TEXAS INSTRUMENTS

TSS400 - Family

User's Guide

Sensor Signal
Processors
1992

User's Guide

TSS400 - Family

1992

TSS400 FAMILY USER'S GUIDE

Part I: TSS400 User's Guide

Part II: TSS400/4 User's Guide

Part III: TSS400(/4) Software User's Guide

Part IV: TSS400 Standard User's Guide

Part I

TSS400 Family User's Guide

Sensor Signal Processors

TSS400 USER'S GUIDE
October 1991

Table of Contents

Purpose and Conventions

List of Figures

Purpose and Conventions

The TSS400 User's Guide is intended to aid the development of TSS400 products by collecting and presenting hardware and software information in a manner that will be quickly accessible to engineers and programmers.

The following substitutes have been adopted because typographical limitations do not allow the use of some conventional symbols:

Symbol	Meaning	Example
<>	"Not equal"	Y <> A
[]	Subscript number	R[0]
^	Exponent	T^2
μsec	Microsecond	
msec	Millisecond	
>XX	Hex-Value	>3F4

1 TSS400

This document describes in detail the functional and electrical characteristics of a 4-bit CMOS sensor signal processor with computing capability. Its key applications are A/D conversions, calculating, controlling and displaying information on an LCD display. The part easily interfaces with sensors and actuators.

Typical applications are:
- temperature measurements: calculating, controlling, warning
- pressure and acceleration measurements
- home appliances
- intelligent keyboard and display driver
- timer with control functions
- intelligent subsystem

1.1 Functional Specification

- 2048 nine bit words of ROM
- 576 bit static RAM
- Three instruction BCD addition
- 64 character, 7 segment output PLA (DP switchable independently)
- Three levels of subroutine
- Timekeeping capability (32 768 Hz XTAL)

- Direct drive of LCD: (32 768 Hz XTAL)
  Option 2MUX: 1/2 duty cycle containing up to 40 segments
  Option 4MUX: 1/4 duty cycle containing up to 80 segments

- Low power silicon gate CMOS process:
  - Low power consumption at sleep mode (active timer/RAM)
  - very low power consumption at off mode (active RAM)

- 12 Bit A/D-Converter with 4-MUX-inputs and programmable ranges
- Programmable current source from 0.15 to 2.4 mA x VDD/V
- Internal MOS oscillator (see Figure 6-1)

- Instruction execution time: 5.5 µs (@f_{proc} = 1.1 MHz)
  15.0 µs (@f_{proc} = 400 KHz)

- Processor frequency determined either by the internal RC-MOS oscillator or the external clock input

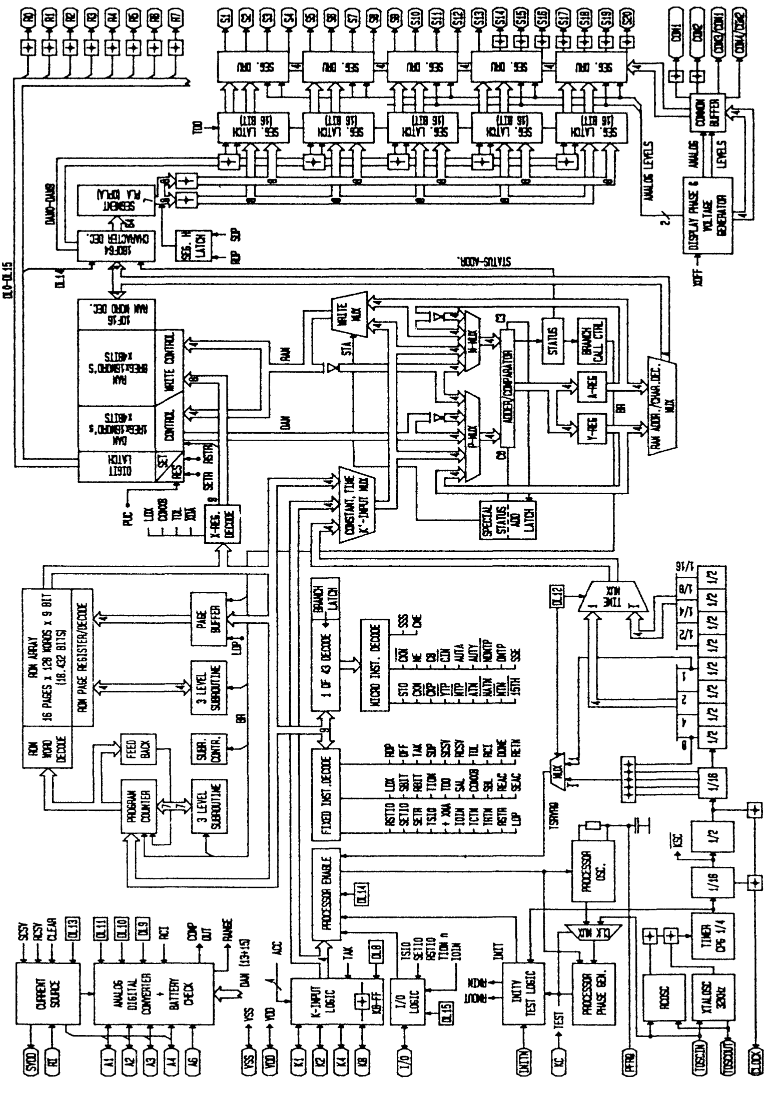


Figure 1-1: Block diagram TSS400

1.2 Memory Structure

The TSS400 has nine (9) registers with sixteen (16) four-bit words per register.

Eight (8) registers are random access memories (RAM) and one (1) register is a direct access memory (DAM).

The 9 registers can be addressed by the X-REG. The content of the X-REG (4 bits) is controlled by the software instructions LDX, TDL, COMX8.

The Y-Register (4 bits) addresses the word within the selected Register. There are several different software instructions that control the contents of the Y-REG.

Since the TSS400 has 9 registers, some values of the X-REG do not address any memory. Figure 1-2 is a list of which values will address memory.

X-REG	HEX	TYPE
0 0 0 1	1	RAM
0 0 1 0	2	RAM
0 0 1 1	3	RAM
0 1 0 0	4	RAM
1 0 0 1	9	RAM
1 0 1 0	A	RAM
1 0 1 1	B	RAM
1 1 0 0	C	RAM
0 1 1 1	7	DAM*
1 1 1 1	F	DAM*

* Both >7 and >F address the same register (the DAM).

Figure 1-2: X-REG Values which Address Memory

In addition to the memory controlled by the X-REG and the Y-REG there are sixteen (16) digit latches which can be set and reset by software instructions. These digit latches are reset by hardware during power-up or by $\overline{\text{INIT}}$. The digit latches are addressed by Y-REG (4 bits) values >0 - >F.

1.3 Subroutine Stack Operation

The Subroutine Stack has the following registers:
1) Page ROM Address Stack Register (3)
2) Program Counter Stack Registers (3)

The interaction of Page Register, Page ROM Address and Stack is shown for the execution of subroutines.

P-Reg	X	X	2	2	2	3	3	4	4	5	5	6	6	6	6	6	6	6	6	6	6
ROM Addr. Reg.	15	15	15	15	15	15	3	3	4	4	5	5	6		4	4	4	4	3	3	3
Sub-routine Reg. A	X	15	15	X	X	X	X	X	3	3	4	4	5	4	3	3	4	3	15	15	15
Sub-routine Reg. B	X	X	X	X	X	X	X	X	X	X	3	3	4	3	15	15	3	15	15	15	15
Sub-routine Reg. C	X	X	X	X	X	X	X	X	X	X	X	X	3	15	15	15	15	15	15	15	15
	PUC	CALL	LDP2	RETN	BR	LDP3	BR*	LDP4	CALL	LDP5	CALL	LDP6	CALL	RETN	RETN	BR	CALL	RETN	RETN	RETN	RETN

* This branch is executed (branch occurs).

Figure 1-3: Subroutine Stack Operation (See Section 2.2 and 2.3 for more details.)

1.4 Digit Latches/R Outputs

<table>
<tr><th>Digit Latch</th><th colspan="2">Hardware Function</th></tr>
<tr><td>DL0-DL7</td><td colspan="2">Stat. outputs R0-R7 → max. 8 R outputs available</td></tr>
<tr><td>DL8</td><td colspan="2">Control of K-Port: DL8 = 0 -- Input
DL8 = 1 -- Output</td></tr>
<tr><td>DL9-DL11</td><td colspan="2">Analog Mux for ADC: A1-A2-A3-A4 or Battery check</td></tr>
<tr><td>DL12</td><td colspan="2">Selection of 1 Hz (DL12=0) or selection of mask programmed frequency for wake-up of the processor. Also selection of time info to CKB-BUS.</td></tr>
<tr><td rowspan="2">DL13</td><td>DL13</td><td>Function</td></tr>
<tr><td>0
1</td><td>constant current off
constant current on</td></tr>
<tr><td>DL14</td><td colspan="2">MSB of character decode and selection of K8 wake-up edge: leading (DL14 = 0) or trailing (DL14 = 1)</td></tr>
<tr><td rowspan="2">DL15</td><td>DL15</td><td>Function</td></tr>
<tr><td>0
1</td><td>wake-up of I/O enabled
wake-up of I/O disabled</td></tr>
</table>

Figure 1-4: Digit Latches/R Outputs

PUC-signal will reset all Digit Latches.

The digit latches are set by the SETR instruction and are reset by the RSTR instruction.

The value of the Y-REG determines which digit latch is affected. The data of the Digit Latches will be retained when the product is in Sleep mode or Off mode as long as the power is maintained.

1.5 DAM Operation

The DAM outputs are transferred into the ALU input regardless the values of the X-REG. This allows operation on data in DAM and one of the 8 memory banks (RAM) at the same time. The DAM instruction is an example

To write into the DAM, the content of the X-REG has to be 0111 or 1111. This can be accomplished by a LDX 7, LDX 15, or a TDL instruction.

If DAM is addressed by the X-REG, all instructions operating on memory will operate on DAM. An example of this is the execution of a TMA instruction while the X-REG contains a 7. This causes the transfer of the DAM into the ACC.

If X-register is 0111 or 1111 instructions which use RAM and DAM will use DAM only. An example of this is the DMEA instr.:

DAM + MEM (= DAM) + SS → ACC   DAM*2 + SS → ACC

1.6 ALU

The ALU (Arithmetic Logic Unit) can perform addition, subtraction (by two's complement) and comparison of 2 four bit numbers. Data from the Y-register, DAM, CKB multiplexer or RAM is selected by the P-multiplexer and added to or compared with data from CKB, Accumulator or RAM selected by the N-multiplexer. The adder-comparator portion of the ALU adds or compares the data. The status will be set or reset depending on the result. Data from the adder can be transferred to the Accumulator, the Y-register, or can be dumped in case the data is needed for comparison only..

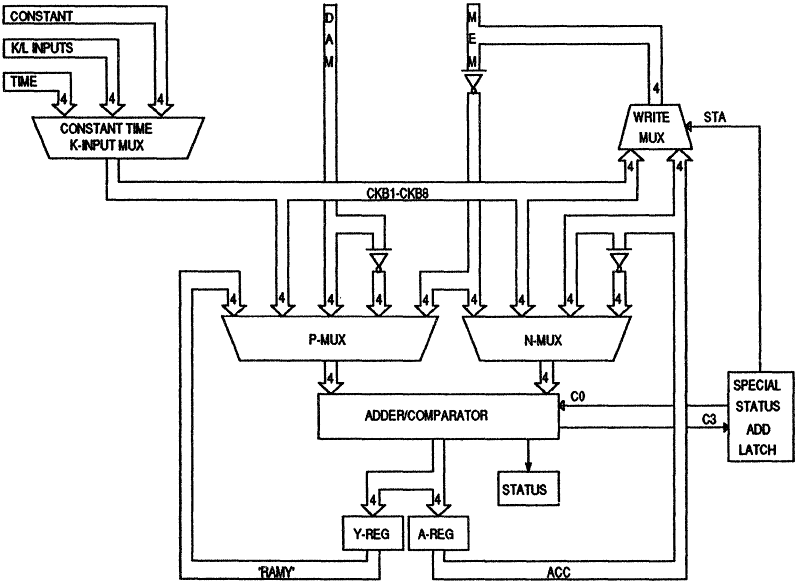


Figure 1-5: Block diagram of ALU

1.7 CKB Memory Map Description

The TSS400 has a 4-bit wide bus line (CKB) that may carry constant data, K-line data, timekeeping data or bit data (SBIT, RBIT, TBIT, TBITA, TCTM, TIOM). Any of these instructions determine which data are valid on the bus line at that time. In order to minimize the physical size of the CKB logic, certain portions of the instruction map are reserved by the CKB logic for data from one (1) of the four (4) sources to be on the CKB bus line.

Instruction Map Locations	Source of Data on CKB Bus
008 → 00F	K-lines
018 → 01F	TKP lines (timekeeping)
020 → 02F	Bit data for TBIT, TCTM
0A0 → 0AF	Bit data for SBIT,RBIT,TBITA,TIOM
040 → 07F	Constant data (I5 → I8)
0C0 → 0FF	Constant data (I5 → I8)

Figure 1-6: Instruction Restrictions

As shown in Figure 1-6, certain areas of the instruction map are decoded by the CKB logic and the CKB bus line will use data from one of its sources. Use micro-instructions CKM, $\overline{\text{CKP}}$, or $\overline{\text{CKN}}$ to transfer data to another portion of the processor. Please note that the source of data on the CKB bus is solely determined by a location in the instruction map as shown above.

1.8 K-Line Structure

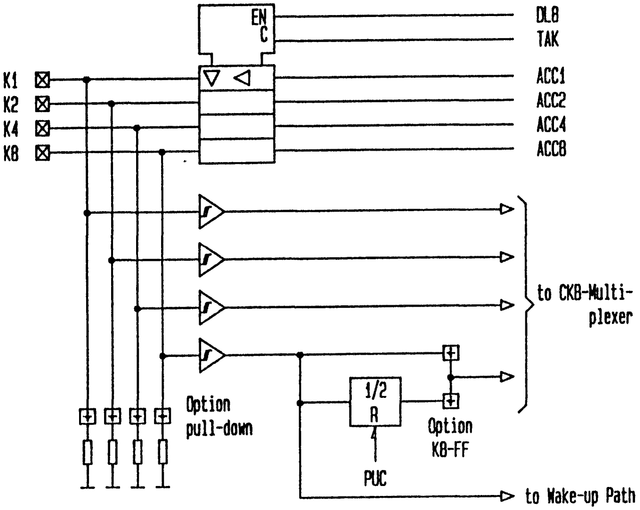


Figure 1-7: K-Line Structure

The present state of Digit latch DL8 defines the state of K-Lines to be Input or Output:

Input if DL8 = "0"
Output if DL8 = "1".

PUC signal will reset DL8 to "0" - Input.

The TAK instruction latches the 4 bit ACC into the output data register. If DL8 is "1" the data is available at the K-Lines:

K1 --- ACC1
K2 --- ACC2
K4 --- ACC4
K8 --- ACC8

The signals, applied at the K-pins, are read or tested directly or divided:

Option K8-FF not used	Option K8-FF used
K1 → ACC1, MEM0	K1 → ACC1, MEM0
K2 → ACC2, MEM1	K2 → ACC2, MEM1
K4 → ACC4, MEM2	K4 → ACC4, MEM2
K8 → ACC8, MEM3	K8/2→ ACC8, MEM3

Three instructions are implemented to read or test the information, applied at the K-Lines:

TKA, TKM and KNEZ.

1.9 I/O Structure

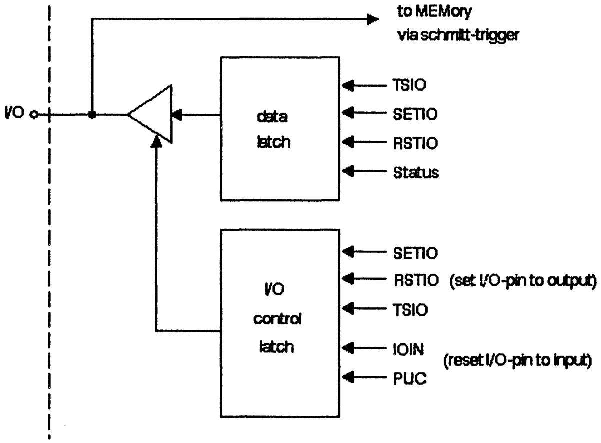


Figure 1-8: I/O Structure

The PUC signal as well as the IOIN instruction will reset the I/O pin to be an input (output buffer is 3-state).

Three instructions define the state of the I/O pin to be an output and will determine the data to be output:
- SETIO : set I/O pin to "1"
- RSTIO : reset I/O pin to "0"
- TSIO : output Status from the previous instruction output.

1.10 Clock

The Clock Output has 3 options that can be selected:

NONE : not selected

256 Hz : $f_{Clock} = \frac{f_{TOSCIN}}{128}$

1024 Hz : $f_{Clock} = \frac{f_{TOSCIN}}{32}$

When the processor is forced into the OFF mode, the oscillator is stopped and the TOSCIN pin is forced to V_{DD} via a p-channel transistor. The signal at the pin 'Clock' remains at the logical level it had during executing the OFF instruction.

When using the Clock Signal together with the OFF mode pay attention to the application S/W and H/W.

If a defined logical level is necessary in the application - e.g. to avoid current consumption - the logical level has to be read e.g. by K-input to check the state of the Clock Signal before executing the OFF instruction.

Note: The signal at the pin 'Clock' is not the same as the signal at the Pin "TOSCIN".

2 Operational Description

There are three different machine states: DONE, ACTIVE and OFF. During the DONE state only display and timekeeping circuitry are active. DONE is a low power state.

In the ACTIVE state the chip is executing instructions from its internal ROM and maximum power is dissipated.

In the OFF state only RAM/DAM and the actual level of I/O, R outputs and K-I/O's remain unchanged. This is true in the DONE state as well.

The machine state transitions are illustrated in Figure 1-12.

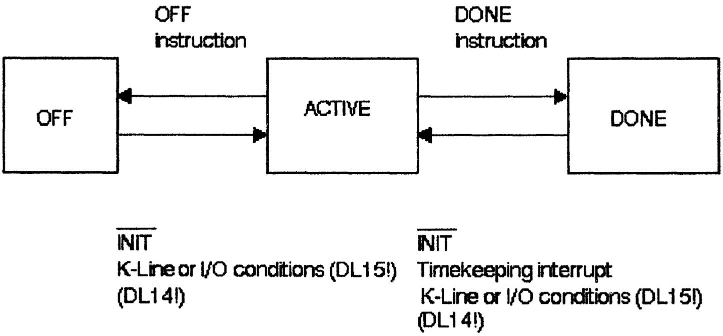


Figure 2-1: Machine Status

2.1 ROM Decode

The ROM on the TSS400 has sixteen (16) pages with 128 words per page and nine bits per word. The ROM page address is contained in the ROM Page Address Register. The ROM word address is derived from the program counter.

The four bits of the page register are decoded in a 1 of 16 decode. A bus runs from the 1 of 16 page decode to the ROM and addresses the proper page.

The seven bits of the program counter are decoded in a 1 of 128 decode. A bus runs from the 1 of 128 word decode to the ROM and addresses one word on each page. Page and word decode select together the particular word.

2.2 Program Counter and Subroutine Register

A bus from the program counter (PC) runs to a 1 of 128 decode and to a feedback network, which is used to change the state of the program counter in order to obtain the next ROM word address. When a CALL instruction is executed, the seven PC bits are stored in the first of a three level subroutine stack. Each successful CALL instruction pushes down the stack. As many CALL instructions as necessary may be executed, but only the three most recent return addresses are stored. All previous return addresses are lost. Each RETN instruction, up to a maximum of three consecutive, pops up the stack. Any RETN instructions encountered when the device is not in a subroutine will be treated as no-op instructions.

2.3 ROM Page Address Register and Page Buffer

The ROM page address register, like the program counter, is capable of three levels of subroutines.

When a CALL instruction is executed, the four page-address bits are stored in the first of a three level subroutine stack. Each successful CALL instruction pushes down the stack. As many CALL instructions as necessary may be executed, but only the three most recent return addresses are stored. All previous addresses are lost. Each RETN instruction, up to a maximum of three consecutive, pops up the stack. Any RETN instruction encountered when the device is not in a subroutine will be treated as a no-op instruction.

When a LDP instruction is executed, the page address is loaded into the Page Buffer. The Page Address is transferred to the ROM Page Address Register when a CALL or BRANCH instruction is successfully executed. If the CALL or BRANCH instruction is unsuccessful or an RETN instruction is executed between the LDP and CALL instructions, the Page Buffer and the ROM Page Address Register remain unchanged.

After an LDP-RETN sequence, the ROM page Address Register remains unchained by CALL and BRANCH instructions.

2.4 Instruction Decode

First a nine-bit word from the ROM goes to the instruction decode, where three types of instructions are decoded:
- programmable instructions
- fixed instructions
- CKB instructions

If 1 of the 48 programmable instructions is decoded as true, a set of micro-instructions will be decoded. These micro-instructions control lines which handle one space too much various logic blocks.

If 1 of the 29 fixed instructions is decoded as true, the output of the respective instruction will set a control line used by a particular logic circuit.

If a CKB instruction is decoded as true, the 4 least significant bits of the ROM word, or the 4 K-lines, or the 4 timekeeping lines are passed into the ALU (depending on which instruction is decoded).

2.5 Processor Enable and Time Base Multiplexer

The software determines when the processor is disabled by executing a DONE instruction. The hardware keeps a timer up to 15 seconds; the software must keep time for all times part 15 seconds (minutes, hours, a.m/p.m., day of the week, date, and month). A calculation must not take longer than 15 seconds. In case a calculation does take longer by software an periodical interrupt of the calculation must occur to enable the timekeeping update.

The processor enable logic "wakes-up" the processor from DONE state depending on DL12, to allow the update of the timekeeping.

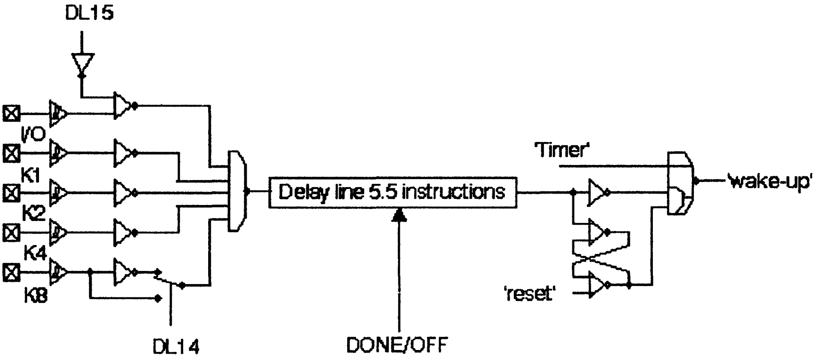
The processor logic-enable-also senses the K-lines K1, K2, K4, K8 and I/O line.

"Wake-up" from OFF state is only possible via Init, K-lines or I/O line; program execution starts at page 15, PC = >7F.

DL14	DL15	K1		K2		K4		K8		I/O	conditions for external wake-up
0	0	0	^	0	^	0	^	0	^	0	before wake-up will be accepted
0	0	↑	v	↑	v	↑	v	↑	v	↑	transition to wake-up processor
1	0	0	^	0	^	0	^	1	^	0	before wake-up will be accepted
1	0	↑	v	↑	v	↑	v	↓	v	↑	transition to wake-up processor
0	1	0	^	0	^	0	^	0	^	X	before wake-up will be accepted
0	1	↑	v	↑	v	↑	v	↑	v	X	transition to wake-up processor
1	1	0	^	0	^	0	^	1	^	X	before wake-up will be accepted
1	1	↑	v	↑	v	↑	v	↓	v	X	transition to wake-up processor

Note: 'v' - logical OR, '^' - logical AND, X - don't care

Figure 2-2: Table for ext. Wake-up Conditions



Note: Wake-up occurs when 'wake-up' signal is Low AND processor in DONE/OFF mode
Delay line is active if processor is active (5 to 6 instructions)
Delay line is inactive if processor is in DONE/OFF mode: No Delay
DL14 selects the logical state of K8 for 'wake-up' signal

Figure 2-3: Schematic of Wake-up Path

2.6 Processor Frequency

The processor frequency is derived from the internal RC-MOS Oscillator. The typical characteristic of this Oscillator is shown in Figure 6-1.

This internal MOS Oscillator frequency can be overwritten by an external Oscillator connected to Pin PFRQ. The frequency range for the external Oscillator is defined in Chapter 6.2 Operating Conditions.

2.7 Timer Oscillator

Crystal Oscillator

A 32.768 Hz signal of the crystal oscillator is fed into the crystal clock generator, which divides the frequency by four and generates four phase clocks. These clocks are divided down through a series of divide-by-two circuits into the frequencies needed for timekeeping functions. A signal KSC is produced by the dividing circuit. This signal is used on the logic of the Common Generator, to generate the signals on the two or four common lines.

RC Oscillator

Using the RC oscillator (OPTION "TRC") instead of the crystal oscillator, the same applies as described in Section 2.7 (see Figure 2-4).

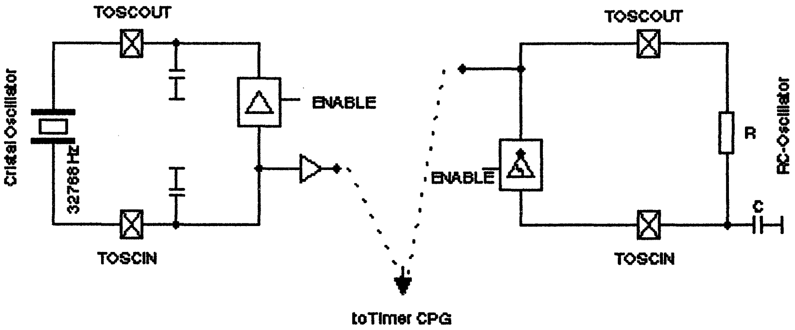


Figure 2-4: Option TIMOSC

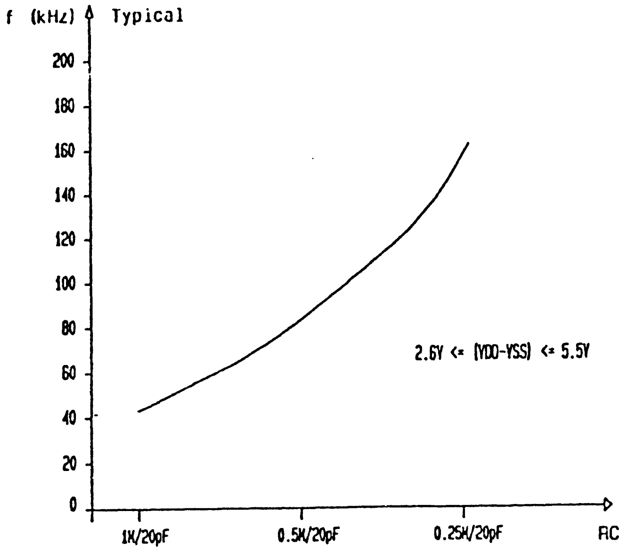


Figure 2-5: TRC Oscillator Frequency

2.8 Time Data Multiplexer

The TSS400 processor does not need to be active all the time. The processor is turned on only when needed to perform a software function or to update timekeeping. The processor can "wake-up" to update timekeeping either once per second or x-times per second (Option).

This is determined in the Time Data Multiplexer. When DL12 is high, the processor wakes-up x*-times per second, and when DL12 is low, the processor wakes-up once per second. This also determines the data used timekeeping.

* Options for x: 128 or 64 or 32 or 16 or 8

When DL12 is high, the 4-bit timekeeping data represents a time of 0/16 to 15/16 of a second, and when DL12 is Low, the data represent a time of 0 to 15 seconds.

The timekeeping can be accessed via CKB-Bus by software.

2.9 CKB (Constant, K-lines, Bit and Time Multiplexer)

All constant data used by the Y-REG, the Accumulator, or memory must pass the CKB. The logic of the CKB determines according to the used instruction whether to select constant, K-lines, bit logic, or timekeeping data to be transferred to various parts of the TSS400.

2.10 Branch/Call Control

When a BRANCH is encountered the Branch/Call Control Logic effects the program counter to dump the present ROM word address and accept a new one. When a CALL is encountered the Branch/Call Logic effects the program counter and the ROM page register to move their present contents into the subroutine latches and accept new addresses. A RETN forces the Branch/Call Control Logic to return the program counter to the stored address plus one. The stored address id the address when the call was encountered. The ROM page address register returns to the address it contained before the CALL. If a RETN is encountered while the CPU is not busy in a subroutine, it is regarded as a no-op and ignored. (See Status paragraph).

2.11 RAM Address/Character Decode Multiplexer

The RAM Address/Character Decode Multiplexer combines 2 blocks:

1) Multiplexer between the Y-Reg and the Accumulator. The output of the multiplexer is fed into the Decoder.
2) Decoder which decodes either the RAM word address from the Y-Register or the LCD character address from the Accumulator.

2.12 Write Multiplexer

The Write Multiplexer determines whether data is to be put into the RAM, the DAM, or the P-Multiplexer in the ALU. The data is taken from the CKB logic or from the Accumulator. The Write Multiplexer is controlled by software.

2.13 X-Decode

The X-REG is decoded by the X-Decode to select a particular RAM bank. Tthe value of the DAM latch, which enables DAM and disables RAM or vice versa is also decoded by the X-REG.

2.14 RAM Word Decode

The RAM Address/Character Decode Multiplexer feeds all 4 bits of the Y-REG into a 1 of 16 decode. The state of the sixteen outputs of the RAM word decode which of the sixteen words in each bank of the RAM and DAM or which of the sixteen Digit latches is addressed.

2.15 Character Decode

The Character Decode uses DL14, status, and the four Accumulator bits to do a 1 of 64 character decode. The 64 signals are then transferred into the output PLA, where they are converted into a seven-segment output. (The Character Decode is illustrated in the Gate Level PLA paragraph.)

2.16 Segment Latches

After decoding and converting the characters into segment outputs, they are latched in Segment latches when the proper Y-pointer is set and a TDO instruction is executed.

2.17 Segment Drivers

The segment data is transferred to the Multiplexing Segment Drivers which convert the seven segments and the Segment H information into 2Mux or 4Mux segment information, which is output and used to turn on the LCD segments.

In test mode some of the LCD-Outputs do have other functions.

2.18 Power Up Clear

There is an time interval between power is applied to the TSS400 and the starting of the clocks. The Power Up Clear (PUC) signal sets the page and word address and enables the PC after the clocks have started.

When the voltage is applied, PUC starts the program execution on page 15, PC = >00. A low signal at $\overline{\text{INIT}}$ pin creates the same function as power being applied.

When the Processor is activated from Done or OFF mode via a K-input-I/O or by a timekeeping update, the program execution begins at Page 15, PC = >7F.

2.19 Status

Status is held in a single bit latch. When this bit is set, which is its normal state, and a BR or CALL is encountered, the BR or CALL will be executed. When status is reset immediately before a CALL or BR, the instruction will not be executed successfully and the Program Counter will continue incrementing.

Status will remain reset (0) for only one instruction cycle.
Status can be unconditionally reset by instruction ACACC 0.
Status can be unconditionally set by instruction ACYY 0.

2.20 P-Multiplexer

The P-Multiplexer part of the ALU selects data from the Y-REG, the CKB, the RAM, or the DAM, and feeds this data to the Adder-Comparator part of the ALU. (See Figure 1-5)

2.21 N-Multiplexer

The N-Multiplexer part of the ALU selects data from the CKB, the Accumulator, or the RAM, and feeds this data to the Adder-Comparator portion of the ALU. (See Figure 1-5)

2.22 Adder-Comparator

The Adder-Comparator portion of the ALU simultaneously adds and compares the data in the N and P Multiplexers. Information from the comparator is used in determining status. Data from the Adder can be sent to the Accumulator or the Y-REG, or can be dumped if the data is needed for comparison only.

2.23 Accumulator and Y-Register

The Accumulator is a four-bit register that gets its content from the Adder-Comparator. The Y-Register (Y-REG) holds the RAM word address and is used to select one of sixteen Digit latches.

3 Analog/Digital-Converter (ADC)

The A/D-Converter is specially designed for:
- 1-2 KOhm-temperature Si-sensors
- Pt100/Pt500/Pt1000 Elements in conjunction with internal current source
- Silicon pressure sensors

It is in no way restricted to those applications.

The A/D-Converter compares the external analogue input voltage with an internal voltage.

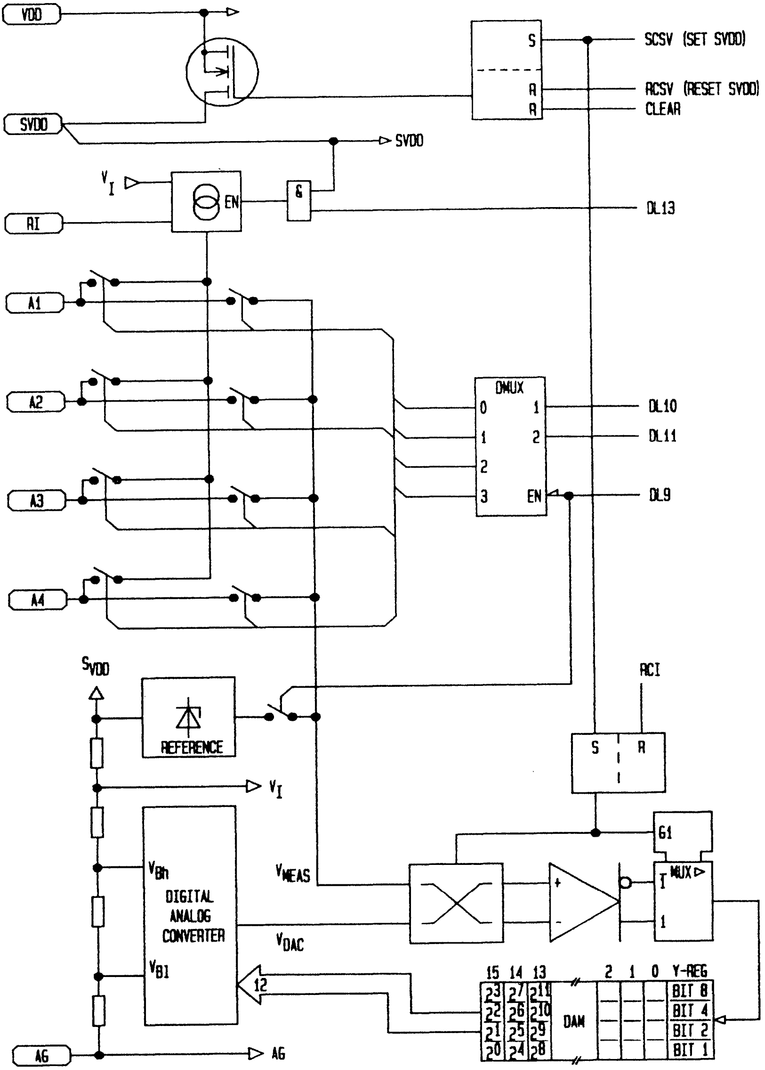


Figure 3-1: Principle of A/D-Converter

3.1 Description of Conversion

The A/D-Conversion is done by executing a software program.

Five instructions are added to the instruction set for controlling the ADC:
- SCSV: switch converter's supply voltage on
- RCSV: reset converter's supply voltage
- RCI: reset converter's comparator inputs to inverted
- TCTM n: transfer comparator (output) to memory - selected by X-REG and Y-REG while n identifies the bit - and set the next lower bit to high (if n is 2,3,4).
- TRTM n: transfer range to memory bit n (n =1,2,3,4).

The A/D-converter consists of:
- a D/A converter (DAC) as voltage reference source
- a Comparator
- an analog switch to interchange input signals at comparator
- an analog multiplexer (4 to 1)
- a constant current source; programmable by one external resistor and enabled or disabled by DL13
- the SV_{DD} control
- a voltage source (for battery check)

As shown in the Figure 3-1, the instruction SCSV switches the supply voltage to the A/D Converter and PIN SV_{DD} on - instruction RCSV switches it off. The voltage is also switched-off during Power on/$\overline{\text{INIT}}$ and if the processor is in DONE/OFF mode.

The D/A converter is controlled by 12 bits. These 12 bits are memory bits located in the DAM.

DAM

Y-REG	BIT3	BIT2	BIT1	BIT0
13	2^{11}	2^{10}	2^{9}	2^{8}
14	2^{7}	2^{6}	2^{5}	2^{4}
15	2^{3}	2^{2}	2^{1}	2^{0}

The content of the 12 DAM bits is linearly converted into an analogue voltage by the DAC. This output voltage is fed into the analogue cross bar switch in front of the comparator. The comparator compares this voltage with the analogue input voltage at the selected input A1 ... A4.

A special instruction TCTM is available to do a simple successive approximation by software. This instruction transfers the comparator output to the pre-selected DAM-bit and sets the next lower bit at the same DAM address.

The starting value should be >800 (midpoint of range). The time constant of the DAC network requires some wait cycles between the TCTM instructions:

affected bit	instructions before TCTM
MSB	5
MSB-1	5
MSB-2	5
MSB-3	5
MSB-4	4
MSB-5	4
MSB-6	4
MSB-7	4
MSB-8	3
MSB-9	3
MSB-10	3
LSB	3

Instruction SCSV sets the analog crossbar switch at the comparator input and instruction RCI resets it. Two conversions - one with set and one with reset switch - eliminate the offset of the comparator. The two conversion results are added. The sum is twice the value of the conversion formula.

To ensure that the measurement is within the valid range, a special instruction TRTM is included. This instruction sets the addressed RAM bit to "0" if the measured value is greater than >000 and less than >FFF. Otherwise the bit is set ("1"): selected analog input voltage out of A/D-conversion range.

Optionally, there are four conversion ranges for the ADC available. One and only one must be selected. This selection is done in connection with the ROM programming.

Conversion formula: (for nominal values)

with: all voltages are referenced to AG

$$\boxed{Vin = (a + k*N) * SVDD} \Rightarrow \boxed{N = \frac{1}{k} * \frac{Vin}{SVDD} - \frac{a}{k}}$$

$$\boxed{\text{with: } 1_{16} \leq N \leq FFE_{16}}$$

single measurement

$$\boxed{\text{with: } N_{LL} \leq N \leq N_{HL}}$$

measurement with interchanged inputs at Comparator inputs (using SCSV / RCI)

Range large	: a = 0.101213203;	k = 0.000096090233 *N
Range small	: a = 0.231271438;	k = 0.000043048228 *N
Range medium-low	: a = 0.1002948411;	k = 0.000073816955 *N
Range medium-high	: a = 0.2326179386;	k = 0.000065411591 *N

Note: N_{LL} = digital value analog input voltage lower limit.
N_{HL} = digital value analog input voltage higher limit.

3.2 Battery Check

The stable internal voltage source (V_{REF}) is applied to the comparator instead of an external voltage signal at the analog inputs. To the other input of the comparator, the DAC output voltage is applied. This DAC output (V_{DAC}) is a linear function of SV_{DD}, that is nearly identical to V_{DD}:

$$V_{DAC} = (a + K \cdot N) \cdot SV_{DD}$$

The comparator allows to determine the N value for which $V_{DAC} = V_{REF}$ and it follows:

$$SV_{DD} = V_{DD} = \frac{V_{REF}}{a \cdot K \cdot N}$$

3.3 Current Source

The current source and the A/D converter are ratiometric. Voltage used for A/D conversion and the reference voltage (V_I) used to set the current of the current source are proportional to SV_{DD} and have a fixed ratio to each other. This ensures optimum tracking. The current source is activated with DL13 set AND with SV_{DD} on; both conditions are necessary to switch on the current source.

The current, passed via an analog input to a sensor, is calculated:

$$I_{An} = \frac{V_{Rext}}{R_{ext}}$$

The current, flowing through a sensor (e.g. resistive temperature sensor), generates a voltage drop at the sensor which is available at the analog input for measurement with the A/D converter:

$$V_{in} = I_{An} \cdot R_{in} \quad \text{with } R_{in} \text{ is sensor's resistance}$$

$$V_{in} = V_{Rext} \cdot \frac{R_{in}}{R_{ext}}$$

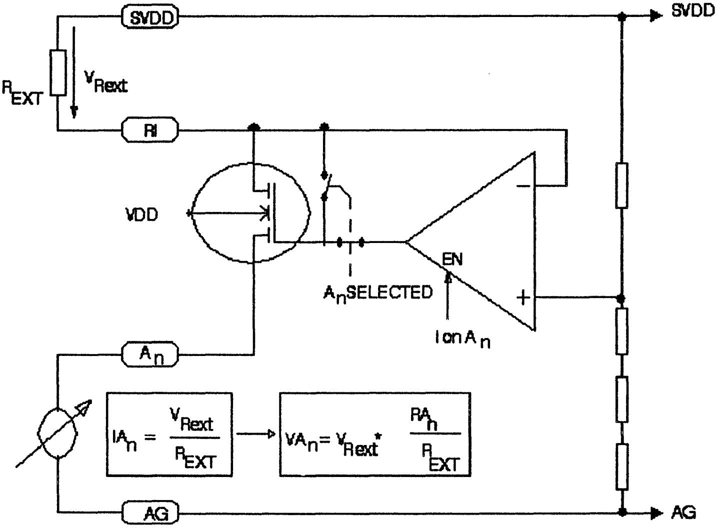


Figure 3-2: Principle of Current Source

DL13	DL11	DL10	DL9	analog input selected, Current Source (CS) on/off
0	0	0	0	A1, CS off
0	0	1	0	A2, CS off
0	1	0	0	A3, CS off
0	1	1	0	A4, CS off
1	0	0	0	A1, CS on
1	0	1	0	A2, CS on
1	1	0	0	A3, CS on
1	1	1	0	A4, CS on
X	X	X	1	battery check, CS off

Figure 3-3: Current Source

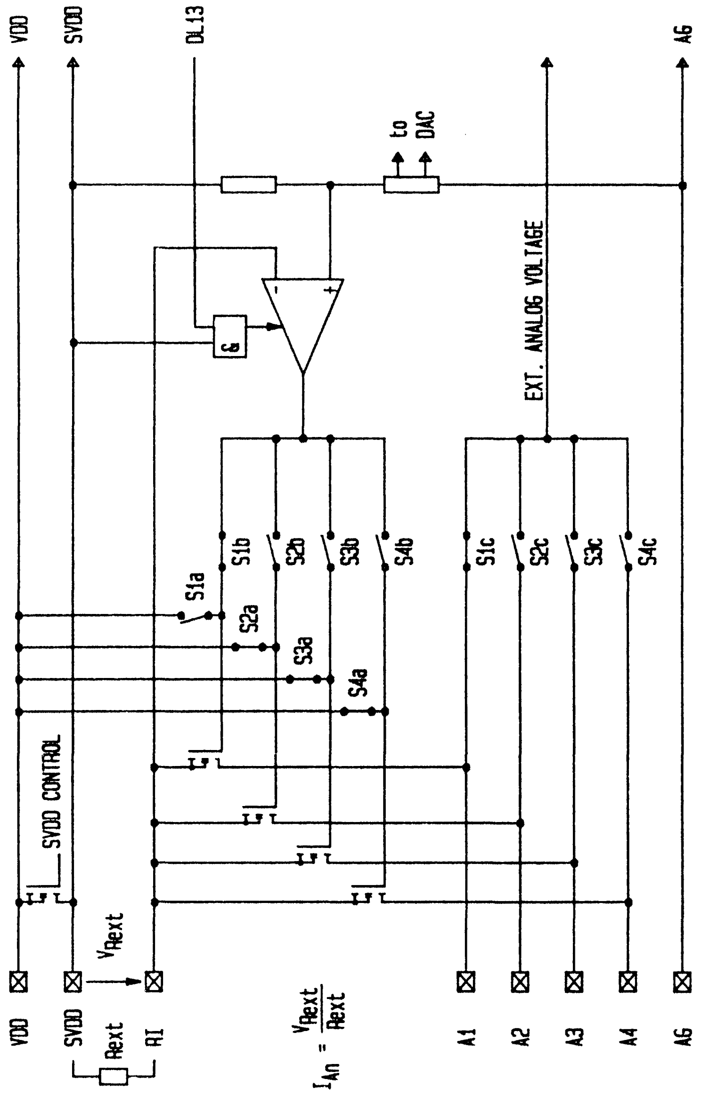


Figure 3-3: Current Source (continued)

4 Display

4.1 Gate Level PLA

The TSS400 has a 1 of 64 decode for character output. The decode has six input lines; from the Most Significant Bit (MSB) to the Least Significant Bit (LSB) they are:

- DL14
- Status bit
- The four Accumulator bits from ACC8 to ACC1.

The decoded information is then transferred to a bus to the Segment PLA. The Segment PLA can be gate-level programmed to set the state of each of the seven segment lines, A-G, high or low depending on the decode information.

The Segment H is controlled by SDP/RDP instruction.

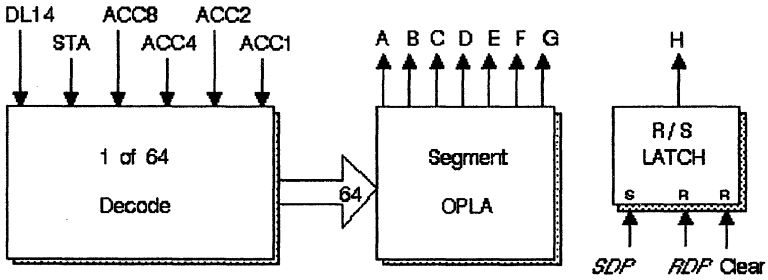


Figure 4-1: Decode and OPLA for Segment Output

4.2 Common Select Lines

There are four Common Select Lines. Each has a unique output waveform that when used in combination with the twenty (20) Select Lines can control 40/80 different segments on a liquid crystal display (LCD).

The combination of Common Select Lines to Segment Lines is shown in the next paragraph.

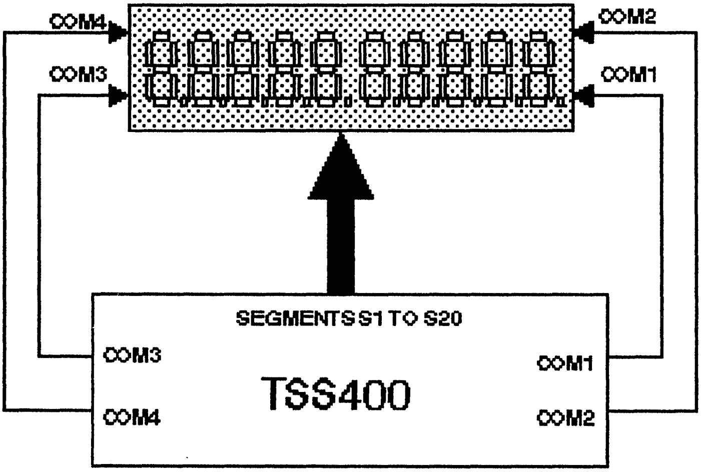


Figure 4-2: 4Mux LCD Application Example

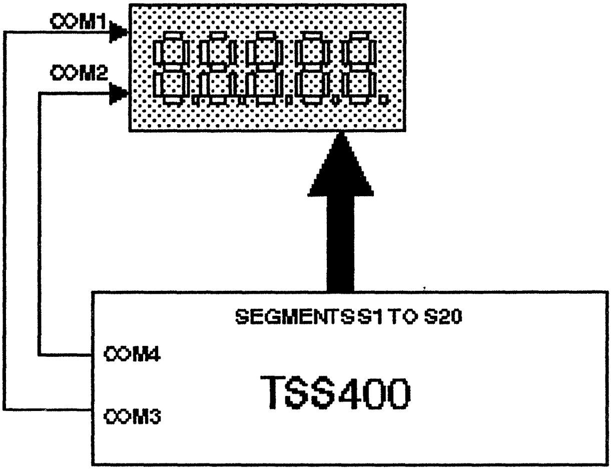


Figure 4-3: 2Mux LCD Application Example

4.3 Select Lines

Up to twenty Select Lines are available as outputs. Each Select Line defines two/four Segments as shown below. A 64-term gate programmable-segment PLA (decoded by ACC1, ACC2, ACC4, ACC8 Status, DL14) and SGH determines which segments are to be turned on.

These are then loaded into the Segment Latches by Y-Decoder/TDO instruction and time-multiplexed to the Output Select Lines.

TSS400 Segment Latch Structure

(Standard Common-Segment-Y option) *).

1/2 duty cycle (2 mux)

Select Line	Segments	Loaded by
	C C O O M M 3 4	
S1	B H	Y0/TDO
S2	A F	Y0/TDO
S3	G E	Y0/TDO
S4	C D	Y0/TDO
S5	B H	Y1/TDO
S6	A F	Y1/TDO
S7	G E	Y1/TDO
S8	C D	Y1/TDO
S9	B H	Y2/TDO
S10	A F	Y2/TDO
S11	G E	Y2/TDO
S12	C D	Y2/TDO
S13	B H	Y3/TDO
S14	A F	Y3/TDO
S15	G E	Y3/TDO
S16	C D	Y3/TDO
S17	B H	Y4/TDO
S18	A F	Y4/TDO
S19	G E	Y4/TDO
S20	C D	Y4/TDO

1/4 duty cycle (4 mux)

Select Line	Segments	Loaded by
	C C C C O O O O M M M M 1 2 3 4	
S1	A B C D	Y0/TDO
S2	F G E H	Y0/TDO
S3	A B C D	Y1/TDO
S4	F G E H	Y1/TDO
S5	A B C D	Y2/TDO
S6	F G E H	Y2/TDO
S7	A B C D	Y3/TDO
S8	F G E H	Y3/TDO
S9	A B C D	Y4/TDO
S10	F G E H	Y4/TDO
S11	A B C D	Y5/TDO
S12	F G E H	Y5/TDO
S13	A B C D	Y6/TDO
S14	F G E H	Y6/TDO
S15	A B C D	Y7/TDO
S16	F G E H	Y7/TDO
S17	A B C D	Y8/TDO
S18	F G E H	Y8/TDO
S19	A B C D	Y9/TDO
S20	F G E H	Y9/TDO

*) Note: The Y-Address of each Segment Group is configurable to application needs. The definition is done together with the ROM programming (via mask).

4.4 Display Drive Waveforms

Display-Drive waveforms are shown in Figure 4-4 for 1/2 and 1/4 duty cycle LCD. The internal logic signal KSC initiates voltage changes in the display voltages.

KSC has a frequency of 512 Hz and is active for 122 micro-seconds (f_{TOSCIN} = 32768 Hz).

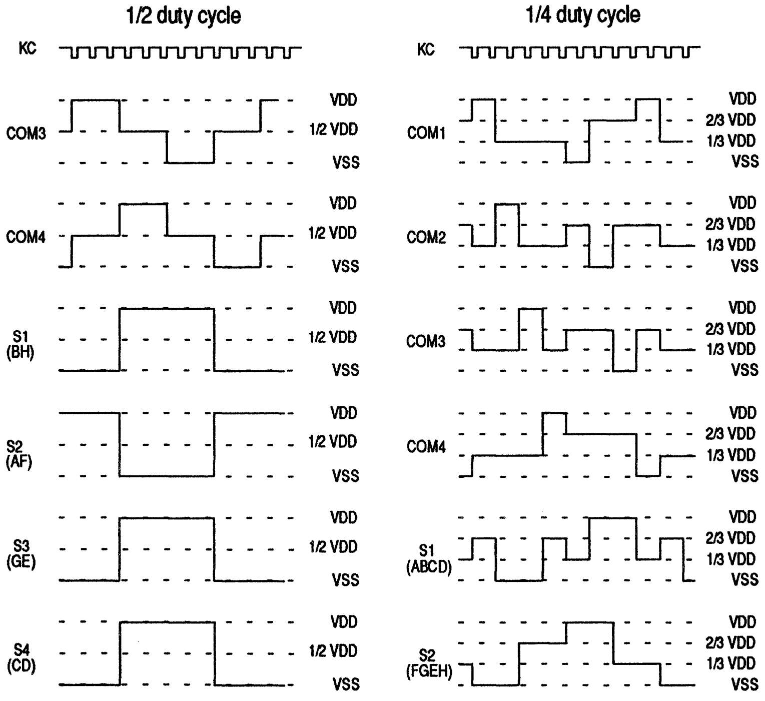


Figure 4-4: Display Drive Waveforms (Standard Gate Placement) e.g. Number 4: Segment 'BCFG' on - 'ADEH' off

5 Instruction Set Description

The following abbreviations will be used in describing the Instruction Set:

ACC	= Accumulator
CONSTANT	= 0-15
LSB	= Least Significant Bit
MEM	= Memory as pointed to by X and Y Registers
MSB	= Most Significant Bit
REG	= Register
SS	= Special Status Latch
<>	= not equal
$\overline{\text{ACC}}$	= not Accumulator

5.1 Functional Description (alphabetical listing)

CODE	MNEMONIC	DESCRIPTION
070-07F	**ACACC**	Add a constant to ACC and store in ACC; if carry status = 1
0F0-0FF	**ACYY**	Add a constant to Y-REG
0E0-EF	**ALEC**	If ACC less than or equal CONSTANT, status = 1
001	**ALEM**	If ACC is less than or equal to MEM, status = 1
015	**AMAAC**	Put MEM + ACC into ACC; if carry, status = 1
100-17F	**BRANCH**	If status = 1, branch to location >00 to >7F as pointed to by label; else increment Program Counter
180-1FF	**CALL**	If status = 1, then execute CALL to location >00 - >7F as defined by label. The 3-level subroutine stack retains the return address.
012	**CCLA**	If SS = 0, put 0 in ACC; if SS = 1, put 1 in ACC
006	**CLA**	Clear ACC
0B2	**COMX8**	Complement MSB of X-REG
031	**CPAIZ**	Put 2's complement of ACC into ACC; if initial ACC = 0, status = 1
018	**CTMDYN**	If both Add Latch and SS are set or both are reset, put ACC into MEM; Decrement Y-REG; if initial Y <> 0, status = 1
007	**DMAN**	Put MEM-1 into ACC; if MEM <> 0, status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
010	**DMEA**	Put DAM + MEM + SS into ACC
011	**DNAA**	Add DAM, $\overline{ACC}$ and SS and store it in ACC
0BE	**DONE**	Turns the processor portion of the chip off
004	**DYN**	Decrement Y-REG; if initial Y <> 0 then status = 1

032	**IMAC**	Store in ACC the result of MEM +1;
		if MEM = 15 then status = 1

01A	**IOIN**	I/O Output buffer becomes 3-state to be Input for ext. signals
005	**IYC**	Increment Y-REG; if initial Y = 15 then status = 1
00E	**KNEZ**	If K-lines <> 0 then status = 1. If status is set to 0 then another KNEZ 6-instruction cycles later must be done; if status = 0 again you can be certain that K-lines = 0.
080-08F	**LDP**	Load Constant into ROM page buffer
090-09F	**LDX**	Load Constant into X-REG. I(5) = LSB, I(8) = MSB. RESET DAM-LATCH
009	**MNEA**	If MEM <> ACC then status = 1
033	**MNEZ**	If MEM <> 0 then status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
013	**NDMEA**	Add MEM, $\overline{\text{DAM}}$ and SS and store it in ACC
0B7	**OFF**	Turns the Processor and Timer off
014	**ORMA**	Logical OR of MEM and $\overline{\text{ACC}}$ stored in ACC
017	**ORMNAA**	Logical OR of MEM and $\overline{\text{ACC}}$ stored in ACC
0A4-0A7	**RBIT**	Reset one of 4 MEM bits as addressed by I(7), I(8) LSB- Bit 2^0 → @I(7), @I(8) Bit 2^1 → I(7), @I(8) Bit 2^2 → @I(7), I(8) MSB- Bit 2^3 → I(7), I(8)
0BD	**RCI**	Reset Comparator offset switch to INVERTED
0BB	**RCSV**	Reset Converters Supply Voltage and Pin SV_{DD}
0B6	**RDP**	Reset Decimal Point (Segment H)
0B4	**REAC**	Reset END-Around-Carry (SS)
0BF	**RETN**	Reset add latch and branch latch. When in CALL, RETN also transfers subroutine register to Program Counter and transfers ROM page register to Page Buffer + Page Address

5.1 **Functional Description** (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
00B	**RSTIO**	Reset I/O pin - I/O Output buffer is enabled
036	**RSTR**	Resets Digit Latch pointed to by Y-REG
0B1	**SAL**	Set ADD latch
030	**SAMAN**	MEM + $\overline{\text{ACC}}$ +1 → ACC; if MEM ≥ ACC then status = 1
0B3	**SBL**	Set Branch Latch
0A0-A3	**SBIT**	Set 1 of 4 MEM bits as addressed by I7 and I8
0BA	**SCSV**	Set Converters Supply Voltage and Pin SV_{DD}. The Comparator offset is switched to NONINVERTED
0B9	**SDP**	Set Decimal Point (Segment H)
0B5	**SEAC**	Set End-Around-Carry (SS)
00C	**SETIO**	Set I/O - I/O Buffer is enabled
00D	**SETR**	Set Digit Latch pointed to by Y-REG
0B8	**TAK**	Transfer ACC to K-latch for Output
03F	**TAM**	Transfer ACC to MEM
0D0-DF	**TAMACS**	Transfer ACC to MEM; Add a constant to ACC and store in ACC; if Add Latch is set and carry from ALU on previous or present instruction is = 1, set SS; reset otherwise; if Add Latch is reset and if carry from ALU on only previous instruction = 1, set SS
03C	**TAMDYN**	Transfer ACC to MEM; decrement Y-REG; if initial Y <> 0, status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
03D	**TAMIYC**	Transfer ACC to MEM; Increment Y-REG, if initial Y = 15, status = 1
03E	**TAMZA**	Transfer ACC to MEM; clear ACC
038	**TAY**	Transfer ACC to Y-REG
020-023	**TBIT**	Test 1 of 4 MEM bits as addressed by I7 and I8. If MEM bit EQ 1, status = 1
0A8-0AB	**TBITA**	Test 1 of 4 ACC bits as addressed by I7 and I8. If ACC bit EQ 1, status = 1
0C0-0CF	**TCA**	Transfer a Constant to ACC
040-04F	**TCY**	Transfer a Constant to Y-REG
060-06F	**TCMIY**	Transfer a Constant to MEM; increment Y-REG
024-027	**TCTM**	Transfer Comparator to 1 of 4 MEM bits as addressed by I7 and I8, the next lower nibble at the same is set.
016	**TDA**	Transfer DAM to ACC independent of X
0BC	**TDL**	Toggle DAM Latch
0B0	**TDO**	Load the Segment Latches with the segment data (A-G) and decimal point into the latches being addressed by Y (0-4/9)
0AC-0AF	**TIOM**	Transfer I/O to 1 of 4 MEM bits as addressed by I7 and I8 (Level sensed during previous instr.)
008	**TKA**	Transfer K-lines to ACC
00A	**TKM**	Transfer K-lines to MEM
039	**TMA**	Transfer MEM to ACC
01D	**TNEA**	If Timer <> ACC, status = 1
01C	**TNEM**	If Timer <> MEM, status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
03A	**TMY**	Transfer MEM to Y-REG
028-02B	**TRTM**	Transfer Range (ADC) to 1 of 4 MEM as addressed by I7 and I8 (ADC in Range: 0; out of Range: 1)
00F	**TSIO**	Transfer Status to I/O - I/O Buffer is enabled
01E	**TTA**	Transfer seconds to ACC if DL12 = 0; transfer 1/16 seconds to ACC if DL12 = 1
01B	**TTM**	Transfer seconds to MEM if DL12 = 0; transfer 1/16 seconds to MEM if DL12 = 1
01F	**TTY**	Transfer seconds to Y-REG if DL12 = 0; transfer 1/16 seconds to Y-REG if DL12 = 1
03B	**TYA**	Transfer Y-REG to ACC
019	**XDA**	Exchange DAM and ACC
003	**XMA**	Exchange MEM and ACC
002	**YNEA**	If Y-REG <> ACC, status = 1
050-05F	**YNEC**	If Y-REG <> Constant, status = 1

5.2 9-Bit Instruction Map

	LSB							
MSB	0	1	2	3	4	5	6	7
00		ALEM	YNEA	XMA	DYN	IYC	CLA	DMAN
01	DMEA	DNAA	CCLA	NDMEA	ORMA	AMAAC	TDA	ORMNAA
02	TBIT				TCTM			
03	SAMAN	CPAIZ	IMAC	MNEZ			RSTR	
04	TCY							
05	YNEC							
06	TCMIY							
07	ACACC							
08	LDP							
09	LDX							
0A	SBIT				RBIT			
0B	TDO	SAL	COMX8	SBL	REAC	SEAC	RDP	OFF
0C	TCA							
0D	TAMACS							
0E	ALEC							
0F	ACYY							
10 to 17	BRANCH							
18 to 1F	CALL							

Note: e.g. SETR : fixed instruction (28)
+ XDA : micro/fixed instruction
other : micro instructions (48)

5.2 9-Bit Instruction Map (continued)

	LSB							
MSB	8	9	A	B	C	D	E	F
00	TKA	MNEA	TKM	RSTIO	SETIO	SETR	KNEZ	TSIO
01	CTMDYN	+ XDA	IOIN	TTM	TNEM	TNEA	TTA	TTY
02	TRM							
03	TAY	TMA	TMY	TYA	TAMDYN	TAMIYC	TAMZA	TAM
04	TCY							
05	YNEC							
06	TCMIY							
07	ACACC							
08	LDP							
09	LDX							
0A	TBITA				TIOM			
0B	TAK	SDP	SCSV	RCVS	TDL	RCI	DONE	RETN
0C	TCA							
0D	TAMACS							
0E	ALEC							
0F	ACYY							
10 to 17	BRANCH							
18 to 1F	CALL							

Note: e.g. SETR : fixed instruction (28)
+ XDA : micro/fixed instruction
other : micro instructions (48)

6 Electrical Description

6.1 Absolute Maximum Ratings

Voltage applied at V_{DD} to V_{SS}	- 0.3 V to + 7.0 V
Voltage applied to any pin (referenced to V_{SS})	V_{SS} - 0.3 V to V_{DD} + 0.3 V
Diode current at any device terminal	+/- 2 mA
Storage Temperature	- 55 °C to 150 °C

6.2 TSS400 - Operating Conditions

Parameter:	**MIN**	**NOM**	**MAX**	**UNITS**
Supply Voltage, V_{DD}	2.6	3.0	5.5	V
Supply Voltage, V_{SS}	0.0	0.0	0.0	V
Operating Temperature				
Range I	-25	27	+85	°C
Range A	-40	27	+125	°C
Timer Frequency (XTAL)		32.768		KHz
*)Timer Frequency (ext)	20		100	KHz
Processor Frequency internal (f_{proc})		700		
Characteristics: see Figure 6-1				
Processor Frequency external,				
PFRQ Pin (f_{proc})	400	700	1000	KHz
**)Instruction execution time	15	8.5	6	µSec
Capacitance of Display:				
any Common			1000	pF
any Segment			200	pF

*) Note: If LCD-display is used be sure that the frequency meets

the limits of display frame rate $\left(DFR = \frac{\text{timer frequency}}{512}\right)$

**) Note: Each instruction needs 6 cycles of the

processor frequency $f_{proc} = t_{instruct} = \frac{6}{f_{proc}}$

6.3 TSS400 - Operating Characteristics

Parameter	Conditions	V_{DD}	Symbol	Min	Nom	Max	Unit
Supply current (into V_{DD})	(excluding external current)						
	Active Mode		$I_{A/D}$				
	with A/D *)						
	T_A = 0 to 80 °C	3V			300	500	µA
	T_A = - 40 to 125 °C	3V			300	500	µA
	T_A = 0 to 80 °C	5V			800	1100	µA
	T_A = - 40 to 125 °C	5V			800	1400	µA
	Active Mode		I_{act}				
	without A/D *)						
	T_A = 0 to 80 °C	3V			80	140	µA
	T_A = - 40 to 125 °C	3V			80	140	µA
	T_A = 0 to 80 °C	5V			400	500	µA
	T_A = - 40 to 125 °C	5V			520	650	µA
	DONE Mode		I_{Done}				
	(Standby) *)						
	T_A = 0 to 80 °C	3V			4	8	µA
	T_A = - 40 to 125 °C	3V			6	9	µA
	T_A = 0 to 80 °C	5V			10	18	µA
	T_A = - 40 to 125 °C	5V			12	20	µA
	OFF Mode (Halt) *)		I_{OFF}				
	T_A = 0 to 80 °C	3V			0.1	1	µA
	T_A = - 40 to 125 °C	3V			0.1	1	µA
	T_A = 0 to 80 °C	5V			0.1	1	µA
	T_A = - 40 to 125 °C	5V			0.1	1	A

*) Current values are for input levels in the range of
0...0.3 V For V_{KL}, V_{IOL};
(V_{DD} - 0.3 V) V_{DD} For V_{KH}, V_{IOH} - all outputs open

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
K, I/Oinputs (Schmitt-Tr)							
V_{T+} (positive going		3V	V_{KH}	1.5		2.0	V
threshold voltage)		5V	V_{KH}	2.7		3.9	V
V_{T-} (negative going		3V	V_{KL}	0.8		1.5	V
threshold voltage)		5V	V_{KL}	1.0		1.9	V
V_{T+} - V_{T-}		3V		0.4		1.4	V
(Hysteresis)		5V		0.8		2.9	V
Option pull-down	V_{IN} = 0.6V	3V	I_{KR}	5	20	15	µA
	V_{IN} = 0.6V	5V	I_{KR}	5	20	15	µA
	V_{IN} = 3V	3V	I_{KR}	30	95	75	µA
	V_{IN} = 5V	5V	I_{KR}	50	160	125	µA
Option no pull-down	V_{IN} = 0V	3.8V/ 5.5V	I_{KR}	- 0.1		0.1	µA
	V_{IN} = V_{DD}	3.8V/ 5.5V	I_{KR}	- 0.1		0.1	µA

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
K as output							
Option push-pull							
"1"	I_{KHmax} = - 0.1 mA	3V	V_{KH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{KHmax} = - 0.3 mA	3V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
"1"	I_{KHmax} = - 0.5 mA	5V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{KLmax} = + 0.3 mA	3V	V_{KL}	V_{SS}		V_{SS} + 0.4	V
"0"	I_{KLmax} = + 0.4 mA	5V	V_{KL}	V_{SS}		V_{SS} + 0.4	V
Option open-source							
"1"	I_{KHmax} = - 0.1 mA	3V	V_{KH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{KHmax} = - 0.3 mA	3V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
"1"	I_{KHmax} = - 0.5 mA	5V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
	V_K = 2.3 V	3V	I_{KOL}	- 30		30	µA
	V_K = 3.8 V	5V	I_{KOL}	- 30		30	µA
I/O as output							
"1"	I_{IOHmax} = - 0.1 mA	3V	V_{IOH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{IOHmax} = - 0.3 mA	3V	V_{IOH}	V_{DD} - 0.6		V_{DD}	V
"1"	I_{IOHmax} = - 0.75mA	5V	V_{IOH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{IOLmax} = + 0.5 mA	3V	V_{IOL}	V_{SS}		V_{SS} + 0.4	V
"0"	I_{IOLmax} = + 1.0 mA	5V	V_{IOL}	V_{SS}		V_{SS} + 0.4	V
R output							
"1"	I_{RHmax} = - 0.1 mA	3V	V_{RH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{RHmax} = - 0.3 mA	3V	V_{RH}	V_{DD} - 0.6		V_{DD}	V
"1"	I_{RHmax} = - 0.3 mA	5V	V_{RH}	V_{DD} - 0.4		V_{DD}	V
"0"	I_{RLmax} = + 0.3 mA	3V	V_{RL}	V_{SS}		V_{SS} + 0.4	V
"0"	I_{RLmax} = + 0.3 mA	5V	V_{RL}	V_{SS}		V_{SS} + 0.4	V

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
Clock (Opt.: NONE, 256Hz, 1024Hz not active at OFF mode)							
output "1"	I_{CH} = - 0.1 mA	3V	V_{CH}	V_{DD} - 0.2		V_{DD}	V
	I_{CH} = - 0.6 mA	5V	V_{CH}	V_{DD} - 0.6		V_{DD}	V
output "0"	I_{CL} = + 0.5 mA	3V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
	I_{CL} = + 0.5 mA	5V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
INITN	V_{IN} = 0V	3V	I_{INIT}	- 0.2		- 1.0	µA
	V_{IN} = 0V	5V	I_{INIT}	- 0.5		- 2.2	µA
	V_{IN} = V_{DD}	3.8V/5.5V	I_{INIT}	- 0.1		0.1	µA
SV_{DD} (switched V_{DD})	I_{SVDD} = 2 mA	2.6V	V_{SVDD}	V_{DD} - 0.2		V_{DD}	V
	I_{SVDD} = 6.5 mA	2.6V	V_{SVDD}	V_{DD} - 0.3		V_{DD}	V
	SV_{DD} off (0V)	3.8V/5.5V	I_{SVDD}	- 0.1		0.1	µA
KC	V_{KC} = 0V	3.8V/5.5V	I_{KC}	- 0.1		0.1	µA
	V_{KC} = V_{DD}	3.8V/5.5V	I_{KC}	- 0.1		0.1	µA

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common (1/2 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
	I_{CH} = - 100 µA	5V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
"1/2 Hz"	I_{CHz} = +/-10 nA	3V	V_{Hz}	V_{DD}/2-.06	V_{DD}/2	V_{DD}/2+.06	V
	I_{CHz} = +/-10 nA	5V	V_{Hz}	V_{DD}/2-.06	V_{DD}/2	V_{DD}/2+.06	V
"1/2 Hz"	V_{CHz} = 0V	3V	I_{CHz}	-1.5		- 4	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	1.5		4	µA
	V_{CHz} = 0V	5V	I_{CHz}	-2.5		- 6.8	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	2.5		6.8	µA
output "0"	I_{CL} = + 50 µA	3V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
	I_{CL} = + 100 µA	5V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
LCD lines segment (1/2 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{SH}	V_{DD} - 0.4		V_{DD}	V
output "1"	I_{CH} = - 100 µA	5V	V_{SH}	V_{DD} - 0.4		V_{DD}	V
output "0"	I_{CL} = + 50 µA	3V	V_{SL}	V_{SS}		V_{SS} + 0.4	V
output "0"	I_{CL} = + 100 µA	5V	V_{SL}	V_{SS}		V_{SS} + 0.4	V

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common,segment (1/4 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
	I_{CH} = - 100 µA	5V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
output "2/3 Hz"	I_{CHz} = +/- 10 nA	3V	V_{Hz}	$2 \cdot V_{DD}/3$ - 0.04	$2 \cdot V_{DD}/3$	$2 \cdot V_{DD}/3$ + 0.04	V
	I_{CHz} = +/- 10 nA	5V	V_{Hz}	$2 \cdot V_{DD}/3$ - 0.04	$2 \cdot V_{DD}/3$	$2 \cdot V_{DD}/3$ + 0.04	V
"2/3 Hz"	V_{CHz} = 0V	3V	I_{CHz}	- 1.5		- 4	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	0.75		2	µA
	V_{CHz} = 0V	5V	I_{CHz}	- 2.5		- 6.8	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	1.25		3.4	µA
output "1/3 Hz"	I_{CHz} = +/- 10 nA	3V	V_{Hz}	$V_{DD}/3$ - 0.04	$V_{DD}/3$	$V_{DD}/3$ + 0.04	V
	I_{CHz} = +/- 10 nA	5V	V_{Hz}	$V_{DD}/3$ - 0.04	$V_{DD}/3$	$V_{DD}/3$ + 0.04	V
"1/3 Hz"	V_{CHz} = 0V	3V	I_{CHz}	- 0.75		- 2	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	1.5		4	µA
	V_{CHz} = 0V	5V	I_{CHz}	-1.25		- 3.4	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	2.5		6.8	µA
output "0"	I_{CL} = + 50 µA	3V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
	I_{CL} = + 100 µA	5V	V_{CL}	V_{SS}		V_{SS} + 0.4	V

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext} = V_{SVDD} - V_{RI}	operating range of V_{DD} is limited to 3.5V ≤ V_{DD} ≤ 5.5V						
Range large*	I_{RI}=1.3 mA ; T_A = 25 °C	3.5V	V_{Rext}	0.240737·SV_{DD}	0.244403·SV_{DD}	0.248069·SV_{DD}	V
	I_{RI}=1.3 mA ; T_A = 25 °C	5V	V_{Rext}	0.241959·SV_{DD}	0.244403·SV_{DD}	0.246847·SV_{DD}	V
external Resistor	T_A = 25 °C	3.5V	R_{ext}	0.10		1.6	kOhm
A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA,						
	AV_{LL}	3.5V	dN/dT		0.03		LSB/°C
	AV_{HL}	3.5V	dN/dT		0.06		LSB/°C
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	5V	dN/dT		0.03		LSB/°C
	AV_{HL}	5V	dN/dT		0.06		LSB/°C

* This range is fully available only for V_{DD} at or above 3.5 V

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3.5V	dN/ dSV_{DD}	- 3	- 1.5	1	LSB/V
	AV_{HL}	3.5V	dN/ dSV_{DD}	- 6	- 3	2	LSB/V
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	5V	dN/ dSV_{DD}	- 3	- 1.5	1	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 6	- 3	2	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3.5V	dV_{Rext}/ month				
	V_{Rext}/R_{ext} = 1.3 mA	5V	dV_{Rext}/ month				

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range large)*							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD} ; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	NLL = 046_{16}	3.5V	AV_{LL}	$0.106017 \cdot SV_{DD}$	$0.107939 \cdot SV_{DD}$	$0.109861 \cdot SV_{DD}$	V
	NLL = $02B_{16}$	5V	AV_{LL}	$0.103423 \cdot SV_{DD}$	$0.105345 \cdot SV_{DD}$	$0.107267 \cdot SV_{DD}$	V
Analog input voltage higher limit	NHL = $FB9_{16}$	3.5V	AV_{HL}	$0.486055 \cdot SV_{DD}$	$0.487977 \cdot SV_{DD}$	0.489899·SVDD	V
	NHL = $FD4_{16}$	5V	AV_{HL}	$0.488649 \cdot SV_{DD}$	$0.490571 \cdot SV_{DD}$	$0.492493 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FB9_{16}$ - 046_{16})	3.5V	R_{AV} (AV_{HL}- AV_{LL})	$0.378596 \cdot SV_{DD}$	$0.380037 \cdot SV_{DD}$	$0.381478 \cdot SV_{DD}$	V
	digital value ($FD4_{16}$ - $02B_{16}$)	5V	R_{AV} (AV_{HL}- AV_{LL})	$0.383784 \cdot SV_{DD}$	$0.385226 \cdot SV_{DD}$	$0.386667 \cdot SV_{DD}$	V

* This range is fully available only for V_{DD} at or above 3.5 V

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3.5V			$96.1 \cdot 10^{-6} \cdot SV_{DD}$		V
		5V			$96.1 \cdot 10^{-6} \cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	3.5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3.5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3.5V		-2 1/2		+2 1/2	LSB
	DDV >2600	3.5V		-4 1/2		+4 1/2	LSB
	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source $V_{Rext} = V_{SVDD} - V_{RI}$							
Range small	I_{RI} = 1.3 mA T_A = 25 °C	3V	V_{Rext}	$0.236635 \cdot SV_{DD}$	$0.240238 \cdot SV_{DD}$	$0.243842 \cdot SV_{DD}$	V
	I_{RI} = 1.3 mA T_A = 25 °C	5V	V_{Rext}	$0.237836 \cdot SV_{DD}$	$0.240238 \cdot SV_{DD}$	$0.242641 \cdot SV_{DD}$	V
external Resistor	T_A = 25 °C	3V	R_{ext}	0.10		1.6	kOhm
A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	3V	dN/dT		0.07		LSB/°C
	AV_{HL}	3V	dN/dT		0.14		LSB/°C
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	5V	dN/dT		0.07		LSB/°C
	AV_{HL}	5V	dN/dT		0.14		LSB/°C

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3V	dN/ dSV_{DD}	- 7	- 3.5	2.5	LSB/V
	AV_{HL}	3V	dN/ dSV_{DD}	- 14	- 7	5	LSB/V
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	5V	dN/ dSV_{DD}	- 7	- 3.5	2.5	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 14	- 7	5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3V	dVR_{ext}/ month				
	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range small)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 09C_{16}$	3V	AV_{LL}	$0.236265 \cdot SV_{DD}$	$0.237986 \cdot SV_{DD}$	$0.239708 \cdot SV_{DD}$	V
	$N_{LL} = 05E_{16}$	5V	AV_{LL}	$0.233596 \cdot SV_{DD}$	$0.235317 \cdot SV_{DD}$	$0.237039 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = F63_{16}$	3V	AV_{HL}	$0.399117 \cdot SV_{DD}$	$0.400839 \cdot SV_{DD}$	$0.402560 \cdot SV_{DD}$	V
	$N_{HL} = FA1_{16}$	5V	AV_{HL}	$0.401786 \cdot SV_{DD}$	$0.403508 \cdot SV_{DD}$	$0.405230 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($F63_{16}$-$09C_{16}$)	3V	R_{AV} (AV_{HL}-AV_{LL})	$0.161990 \cdot SV_{DD}$	$0.162851 \cdot SV_{DD}$	$0.163712 \cdot SV_{DD}$	V
	digital value ($FA1_{16}$ - $05E_{16}$)	5V	R_{AV} (AV_{HL}-AV_{LL})	$0.167328 \cdot SV_{DD}$	$0.168189 \cdot SV_{DD}$	$0.169050 \cdot SV_{DD}$	V

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 TSS400 - Operating Characteristics (continued):
T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3V			$43.0 \cdot 10\text{-}6 \cdot SV_{DD}$		V
		5V			$43.0 \cdot 10\text{-}6 \cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	3V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3V		-2 1/2		+2 1/2	LSB
	DDV >2600	3V		-4 1/2		+4 1/2	LSB
	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source							
V_{Rext}= V_{SVDD} - V_{RI}	operating range of V_{DD} is limited to $3.5V \le V_{DD} \le 5.5V$						
Range medium-high*	I_{RI}=1.3 mA ; T_A = 25 °C	3.5V	V_{Rext}	$0.237979 \cdot SV_{DD}$	$0.241603 \cdot SV_{DD}$	$0.245227 \cdot SV_{DD}$	V
	I_{RI}=1.3 mA ; T_A = 25 °C	5V	V_{Rext}	$0.239187 \cdot SV_{DD}$	$0.241603 \cdot SV_{DD}$	$0.244019 \cdot SV_{DD}$	V
external Resistor	T_A = 25 °C	3.5V	R_{ext}	0.10		1.6	kOhm
A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	3.5V	dN/dT		0.04		LSB/°C
	AV_{HL}	3.5V	dN/dT		0.08		LSB/°C
Temperature stability	V_{Rext}/R_{ext} =1.3 mA, AV_{LL}	5V	dN/dT		0.04		LSB/°C
	AV_{HL}	5V	dN/dT		0.08		LSB/°C

* This range is fully available only for V_{DD} at or above 3.5 V

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3.5V	dN/ dSV_{DD}	- 4.5	- 2	1.5	LSB/V
	AV_{HL}	3.5V	dN/ dSV_{DD}	- 9	- 4.5	3	LSB/V
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	5V	dN/ dSV_{DD}	- 4.5	- 2	1.5	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 9	- 4.5	3	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3.5V	dVR_{ext}/ month				
	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-high)*							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD} ; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage	$N_{LL} = 067_{16}$	3.5V	AV_{LL}	$0.237393 \cdot SV_{DD}$	$0.239355 \cdot SV_{DD}$	$0.241317 \cdot SV_{DD}$	V
lower limit	$N_{LL} = 03E_{16}$	5V	AV_{LL}	$0.234711 \cdot SV_{DD}$	$0.236673 \cdot SV_{DD}$	$0.238635 \cdot SV_{DD}$	V
Analog input voltage	$N_{HL} = F98_{16}$	3.5V	AV_{HL}	$0.491765 \cdot SV_{DD}$	$0.493731 \cdot SV_{DD}$	$0.495694 \cdot SV_{DD}$	V
higher limit	$N_{HL} = FC1_{16}$	5V	AV_{HL}	$0.494451 \cdot SV_{DD}$	$0.496413 \cdot SV_{DD}$	$0.498375 \cdot SV_{D}$	V
Range analog input volt. (see note)	digital value ($F98_{16}$ - 067_{16})	3.5V	R_{AV} (AV_{HL} - AV_{LL})	$0.253405 \cdot SV_{DD}$	$0.254386 \cdot SV_{DD}$	$0.255367 \cdot SV_{DD}$	V
	digital value ($FC1_{16}$ - $03E_{16}$)	5V	R_{AV} (AV_{HL} - AV_{LL})	$0.258768 \cdot SV_{DD}$	$0.259750 \cdot SV_{DD}$	$0.260731 \cdot SV_{DD}$	V

* This range is fully available only for V_{DD} at or above 3.5 V

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3.5V			$65.4 \cdot 10^{-6} \cdot SV_{DD}$		
		5V			$65.4 \cdot 10^{-6} \cdot SV_{DD}$		
Linearity	Delta digital value ≤ 120 LSB	3.5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3.5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3.5V		-2 1/2		+2 1/2	LSB
	DDV >2600	3.5V		-4 1/2		+4 1/2	LSB
	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.3 **TSS400 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext} = V_{SVDD} - V_{RI}							
Range medium-low	I_{RI} = 1.3 mA T_A = 25 °C	3V	V_{Rext}	0.238629·SV_{DD}	0.242263·SV_{DD}	0.245897·SV_{DD}	V
	I_{RI} = 1.3 mA T_A = 25 °C	5V	V_{Rext}	0.239841·SV_{DD}	0.242263·SV_{DD}	0.244686·SV_{DD}	V
external Resistor	T_A = 25 °C	3V	R_{ext}	0.10		1.6	kOhm
A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	3V	dN/dT		0.04		LSB/°C
	AV_{HL}	3V	dN/dT		0.08		LSB/°C
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	5V	dN/dT		0.04		LSB/°C
	AV_{HL}	5V	dN/dT		0.08		LSB/°C

6.3 TSS400 - Operating Characteristics (continued):

T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3V	dN/ dSV_{DD}	- 4	- 2	1.5	LSB/V
	AV_{HL}	3V	dN/ dSV_{DD}	- 8	- 4	2.5	LSB/V
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	5V	dN/ dSV_{DD}	- 4	- 2	1.5	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 8	- 4	2.5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3V	dVR_{ext}/ month				
	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-low)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage	$N_{LL} = 05B_{16}$	3V	AV_{LL}	$0.104798 \cdot SV_{DD}$	$0.107013 \cdot SV_{DD}$	$0.109227 \cdot SV_{DD}$	V
Lower limit	$N_{LL} = 036_{16}$	5V	AV_{LL}	0.102066·SVDD	$0.104281 \cdot SV_{DD}$	$0.106495 \cdot SV_{DD}$	V
Analog input voltage	$N_{HL} = FA4_{16}$	3V	AV_{HL}	$0.393642 \cdot SV_{DD}$	$0.395857 \cdot SV_{DD}$	$0.398072 \cdot SV_{DD}$	V
Higher limit	$N_{HL} = FC7_{16}$	5V	AV_{HL}	$0.396227 \cdot SV_{DD}$	$0.398441 \cdot SV_{DD}$	$0.400656 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FA4_{16}$ - $05B_{16}$)	3V	R_{AV} (AV_{HL}- AV_{LL})	$0.287739 \cdot SV_{DD}$	$0.288846 \cdot SV_{DD}$	$0.289953 \cdot SV_{DD}$	V
	digital value ($FC7_{16}$ - 036_{16})	5V	R_{AV} (AV_{HL}- AV_{LL})	$0.293053 \cdot SV_{DD}$	$0.294161 \cdot SV_{DD}$	$0.295268 \cdot SV_{DD}$	V

6.3 TSS400 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3V			73.8·10-6·SV_{DD}		V
		5V			73.8·10-6·SV_{DD}		V
Linearity	Delta digital value ≤ 120 LSB	3V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3V		-2 1/2		+2 1/2	LSB
	DDV >2600	3V		-4 1/2		+4 1/2	LSB
	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB
Battery check							
Range small, medium-high	000_{16}<conversion result>FFF_{16}		V_{DDBC}		2.7	3.7	V
Range large, medium-low	000_{16}<conversion result>FFF_{16}		V_{DDBC}		2.7	4.8	V
Drift	N of ADC @ V_{DD} = 2.7V, 25°C			-0.1		+0.1	V

6.4 TSS403 - Operating Conditions

Parameter:	**MIN**	**NOM**	**MAX**	**UNITS**
Supply Voltage, V_{DD}	2.6	3.0	3.8	V
Supply Voltage, V_{SS}	0.0	0.0	0.0	V
Operating Temperature				
Range I	-25	27	+85	°C
Range A	-40	27	+125	°C
Timer Frequency (XTAL)		32.768		KHz
*)Timer Frequency (ext)	20		100	KHz
Processor Frequency internal (f_{proc})		700		
Characteristics : see Figure 6-1				
Processor Frequency external,				
PFRQ Pin (f_{proc})	400	700	1000	KHz
**)Instruction execution time	15	8.5	6	µSec
Capacitance of Display:				
any Common			1000	pF
any Segment			200	pF

*) Note: If LCD-display is used be sure that the frequency meets

the limits of display frame rate $\left(DFR = \frac{\text{timer frequency}}{512}\right)$

**) Note: Each instruction needs 6 cycles of the

processor frequency $f_{proc} = t_{instruct} = \frac{6}{f_{proc}}$

6.5 TSS403 - Operating Characteristics

Parameter	Conditions	V_{DD}	Symbol	Min	Nom	Max	Unit
Supply current (into V_{DD})	(excluding external current)						
	Active Mode		$I_{A/D}$				
	with A/D *)						
	T_A= 0 to 80 °C	3V			300	500	µA
	T_A= -40 to 125 °C	3V			300	500	µA
	Active Mode		I_{act}				
	without A/D *)						
	T_A= 0 to 80 °C	3V			80	140	µA
	T_A= -40 to 125 °C	3V			80	140	µA
	DONE Mode		I_{Done}				
	(Standby) *)						
	T_A= 0 to 80 °C	3V			4	8	µA
	T_A= -40 to 125 °C	3V			6	9	µA
	OFF Mode (Halt) *)		I_{OFF}				
	T_A= 0 to 80 °C	3V			0.1	1	µA
	T_A= -40 to 125 °C	3V			0.1	1	µA

*) Current values are for input levels in the range of
0...0.3 V For V_{KL}, V_{IOL};
(V_{DD} - 0.3 V) V_{DD} For V_{KH}, V_{IOH} - All outputs open

6.5 **TSS403 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
K, I/Oinputs (Schmitt-Tr)							
V_{T+} (positive going threshold voltage)		3V	V_{KH}	1.5		2.0	V
V_{T-} (negative going threshold voltage)		3V	V_{KL}	0.8		1.5	V
V_{T+} - V_{T-} (Hysteresis)		3V		0.4		1.4	V
Option pull-down	V_{IN} = 0.6V	3V	I_{KR}	5	20	15	µA
	V_{IN} = 3V	3V	I_{KR}	30	95	75	µA
Option no pull-down	V_{IN} = 0V	3.8V	I_{KR}	-0.1		0.1	µA
	V_{IN} = V_{DD}	3.8V	I_{KR}	-0.1		0.1	µA

6.5 TSS403 - Operating Characteristics (continued):

T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
K as output							
Option push-pull							
"1"	I_{KHmax} = - 0.1 mA	3V	V_{KH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{KHmax} = - 0.3 mA	3V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{KLmax} = + 0.3 mA	3V	V_{KL}	V_{SS}		V_{SS} + 0.4	V
Option open-source							
"1"	I_{KHmax} = - 0.1 mA	3V	V_{KH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{KHmax} = - 0.3 mA	3V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
	V_K = 2.3 V	3V	I_{KOL}	- 30		30	nA
I/O as output							
"1"	I_{IOHmax} = - 0.1 mA	3V	V_{IOH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{IOHmax} = - 0.3 mA	3V	V_{IOH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{IOLmax} = + 0.5 mA	3V	V_{IOL}	V_{SS}		V_{SS} + 0.4	V
R output							
"1"	I_{RHmax} = - 0.1 mA	3V	V_{RH}	V_{DD} - 0.2		V_{DD}	V
"1"	I_{RHmax} = - 0.3 mA	3V	V_{RH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{RLmax} = + 0.3 mA	3V	V_{RL}	V_{SS}		V_{SS} + 0.4	V

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
Clock (Opt.:NONE,256Hz, 1024Hz) (not active at OFF mode)							
output "1"	I_{CH} = - 0.1 mA	3V	V_{CH}	V_{DD} - 0.2		V_{DD}	V
output "0"	I_{CL} = + 0.5 mA	3V	V_{CL}	V_{SS}		VSS + 0.4	V
INITN	V_{IN} = 0V	3V	I_{INIT}	- 0.2		- 1.0	µA
	V_{IN} = V_{DD}	3.8V	I_{INIT}	- 0.1		0.1	µA
SV_{DD} (switched V_{DD})	I_{SVDD} = 2 mA	2.6V	V_{SVDD}	V_{DD} - 0.2		V_{DD}	V
	I_{SVDD} = 6.5 mA	2.6V	V_{SVDD}	V_{DD} - 0.3		V_{DD}	
	SV_{DD} off (0V)	3.8V	I_{SVDD}	- 0.1		0.1	µA
KC	V_{KC} = 0V	3.8V	I_{KC}	- 0.1		0.1	µA
	V_{KC} = V_{DD}	3.8V	I_{KC}	- 0.1		0.1	µA

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common (1/2 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
"1/2 Hz"	I_{CHz} = +/-10 nA	3V	V_{Hz}	V_{DD}/2-.06	V_{DD}/2	V_{DD}/2+.06	V
"1/2 Hz"	V_{CHz} = 0V	3V	I_{CHz}	-1.5		- 4	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	1.5		4	µA
output "0"	I_{CL} = + 50 µA	3V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
LCD lines segment (1/2 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{SH}	V_{DD} - 0.4		V_{DD}	V
output "0"	I_{CL} = + 50 µA	3V	V_{SL}	V_{SS}		V_{SS} + 0.4	V

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common, segment (1/4 duty cycle)							
output "1"	I_{CH} = - 50 µA	3V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
output "2/3 Hz"	I_{CHz} = +/- 10 nA	3V	V_{Hz}	2·V_{DD}/3-0.04	2·V_{DD}/3	2·V_{DD}/3+0.04	V
"2/3 Hz"	V_{CHz} = 0V	3V	I_{CHz}	- 1.5		- 4	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	0.75		2	µA
output "1/3 Hz"	I_{CHz} = +/- 10 nA	3V	V_{Hz}	V_{DD}/3-0.04	V_{DD}/3	V_{DD}/3+0.04	V
"1/3 Hz"	V_{CHz} = 0V	3V	I_{CHz}	- 0.75		- 2	µA
	V_{CHz} = V_{DD}	3V	I_{CHz}	1.5		4	µA
output "0"	I_{CL} = + 50 µA	3V	V_{CL}	V_{SS}		V_{SS} + 0.4	V

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext}= V_{SVDD} - V_{RI}	operating range of V_{DD} is limited to 3.5V ≤ V_{DD} ≤ 5.5V						
Range large*	I_{RI}=1.3 mA ; T_A = 25 °C	3.5V	V_{Rext}	0.240737·SV_{DD}	0.244403·SV_{DD}	0.248069·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	3.5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA,						
	AV_{LL}	3.5V	dN/dT		0.03		LSB/°C
	AV_{HL}	3.5V	dN/dT		0.06		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3.5V	dN/ dSV_{DD}	- 3	- 1.5	1	LSB/V
	AV_{HL}	3.5V	dN/ dSV_{DD}	- 6	- 3	2	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3.5V	dV_{Rext}/ month				

* This range is fully available only for V_{DD} at or above 3.5 V

6.5 **TSS405 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range large)*							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	NLL = 046_{16}	3.5V	AV_{LL}	$0.106017 \cdot SV_{DD}$	$0.107939 \cdot SV_{DD}$	$0.109861 \cdot SV_{DD}$	V
Analog input voltage higher limit	NHL = $FB9_{16}$	3.5V	AV_{HL}	$0.486055 \cdot SV_{DD}$	$0.487977 \cdot SV_{DD}$	0.489899·SVDD	V
Range analog input volt. (see Note)	digital value ($FB9_{16}$ - 046_{16})	3.5V	R_{AV} (AV_{HL} - AV_{LL})	$0.378596 \cdot SV_{DD}$	$0.380037 \cdot SV_{DD}$	$0.381478 \cdot SV_{DD}$	V

* This range is fully available only for V_{DD} at or above 3.5 V

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.5 **TSS405 - Operating Characteristics** (continued):

T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3.5V			$96.1 \cdot 10^{-6} \cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	3.5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3.5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3.5V		-2 1/2		+2 1/2	LSB
	DDV >2600	3.5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source $V_{Rext} = V_{SVDD} - V_{RI}$							
Range small	I_{RI} = 1.3 mA; T_A = 25 °C	3V	V_{Rext}	0.236635·SV_{DD}	0.240238·SV_{DD}	0.243842·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	3V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA, AV_{LL}	3V	dN/dT		0.07		LSB/°C
	AV_{HL}	3V	dN/dT		0.14		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C AV_{LL}	3V	dN/ dSV_{DD}	- 7	- 3.5	2.5	LSB/V
	AV_{HL}	3V	dN/ dSV_{DD}	- 14	- 7	5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3V	dVR_{ext}/ month				

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range small)							
Input current (any inp.)	V_{IN} = V_{SS} to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage							
lower limit	N_{LL} = $09C_{16}$	3V	AV_{LL}	0.236265·SV_{DD}	0.237986·SV_{DD}	0.239708·SV_{DD}	V
Analog input voltage							
higher limit	N_{HL} = $F63_{16}$	3V	AV_{HL}	0.399117·SV_{DD}	0.400839·SV_{DD}	0.402560·SV_{DD}	V
Range analog input volt. (see Note)	digital value ($F63_{16}$ - $09C_{16}$)	3V	R_{AV} (AV_{HL}- AV_{LL})	0.161990·SV_{DD}	0.162851·SV_{DD}	0.163712·SV_{DD}	V

Note: AD-range limited due to offset of comparator. Nevertheless, the range can be larger by smaller offset.

6.5 **TSS403 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3V			$43.0\cdot 10\text{-}6\cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	3V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3V		-2 1/2		+2 1/2	LSB
	DDV >2600	3V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comparator. Nevertheless, the range can be larger by smaller offset.

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext}= V_{SVDD} - V_{RI}	operating range of V_{DD} is limited to 3.5V ≤ V_{DD} ≤ 5.5V						
Range medium-high*	I_{RI}=1.3 mA ; T_A = 25 °C	3.5V	V_{Rext}	0.237979·SV_{DD}	0.241603·SV_{DD}	0.245227·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	3.5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	3.5V	dN/dT		0.04		LSB/°C
	AV_{HL}	3.5V	dN/dT		0.08		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C, AV_{LL}	3.5V	dN/ dSV_{DD}	- 4.5	- 2	1.5	LSB/V
	AV_{HL}	3.5V	dN/ dSV_{DD}	- 9	- 4.5	3	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3.5V	dVR_{ext}/ month				

* This range is fully available only for V_{DD} at or above 3.5 V

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-high)*							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD} ; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 067_{16}$	3.5V	AV_{LL}	$0.237393 \cdot SV_{DD}$	$0.239355 \cdot SV_{DD}$	$0.241317 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = F98_{16}$	3.5V	AV_{HL}	$0.491765 \cdot SV_{DD}$	$0.493731 \cdot SV_{DD}$	$0.495694 \cdot SV_{DD}$	V
Range analog input volt. (see note)	digital value $(F98_{16} - 067_{16})$	3.5V	R_{AV} $(AV_{HL} - AV_{LL})$	$0.253405 \cdot SV_{DD}$	$0.254386 \cdot SV_{DD}$	$0.255367 \cdot SV_{DD}$	V

* This range is fully available only for V_{DD} at or above 3.5 V

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.5 **TSS403 - Operating Characteristics** (continued):

T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3.5V			$65.4 \cdot 10^{-6} \cdot SV_{DD}$		
Linearity	Delta digital value ≤ 120 LSB	3.5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3.5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3.5V		-2 1/2		+2 1/2	LSB
	DDV >2600	3.5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comperator. Nevertheless the range can be larger by smaller offset.

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext}= V_{SVDD} - V_{RI}							
Range medium-low	I_{RI} = 1.3 mA; T_A = 25 °C	3V	V_{Rext}	0.238629·SV_{DD}	0.242263·SV_{DD}	0.245897·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	3V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA,						
	AV_{LL}	3V	dN/dT		0.04		LSB/°C
	AV_{HL}	3V	dN/dT		0.08		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C,						
	AV_{LL}	3V	dN/ dSV_{DD}	- 4	- 2	1.5	LSB/V
	AV_{HL}	3V	dN/ dSV_{DD}	- 8	- 4	2.5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	3V	dVR_{ext}/ month				

6.5 TSS403 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-low)							
Input current (any inp.)	V_{IN} = V_{SS} to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	N_{LL} = $05B_{16}$	3V	AV_{LL}	$0.104798 \cdot SV_{DD}$	$0.107013 \cdot SV_{DD}$	$0.109227 \cdot SV_{DD}$	V
Analog input voltage higher limit	N_{HL} = $FA4_{16}$	3V	AV_{HL}	$0.393642 \cdot SV_{DD}$	$0.395857 \cdot SV_{DD}$	$0.398072 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FA4_{16}$ - $05B_{16}$)	3V	R_{AV} (AV_{HL}- AV_{LL})	$0.287739 \cdot SV_{DD}$	$0.288846 \cdot SV_{DD}$	$0.289953 \cdot SV_{DD}$	V

6.5 TSS403 - Operating Characteristics (continued):
T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		3V			$73.8 \cdot 10^{-6} \cdot SV_{DD}$		
Linearity	Delta digital value ≤ 120 LSB	3V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	3V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	3V		-2 1/2		+2 1/2	LSB
	DDV >2600	3V		-4 1/2		+4 1/2	LSB
Battery check							
Range small, medium-high	000_{16}<conversion result>FFF_{16}		V_{DDBC}		2.7	3.7	V
Range large, medium-low	000_{16}<conversion result>FFF_{16}		V_{DDBC}		2.7	3.8	V
Drift	N of ADC @ V_{DD}=2.7V,25 °C			-0.1		+0.1	V

6.6 TSS405 - Operating Conditions

Parameter:	**MIN**	**NOM**	**MAX**	**UNITS**
Supply Voltage, V_{DD}	4.5	5.0	5.5	V
Supply Voltage, V_{SS}	0.0	0.0	0.0	V
Operating Temperature				
Range I	-25	27	+85	°C
Range A	-40	27	+125	°C
Timer Frequency (XTAL)		32.768		KHz
*) Timer Frequency (ext)	20		100	KHz
Processor Frequency internal (f_{proc})		700		
Characteristics : see Figure 6-1				
Processor Frequency external,				
PFRQ Pin (f_{proc})	400	700	1000	KHz
**) Instruction execution time	15	8.5	6	µSec
Capacitance of Display:				
any Common			1000	pF
any Segment			200	pF

*) Note: If LCD-display is used be sure that the frequency meets

the limits of display frame rate $\left(DFR = \frac{\text{timer frequency}}{512} \right)$

**) Note: Each instruction needs 6 cycles of the

processor frequency f_{proc} = $t_{instruct} = \frac{6}{f_{proc}}$

6.7 TSS405 - Operating Characteristics

Parameter	Conditions	V_{DD}	Symbol	Min	Nom	Max	Unit
Supply current (into V_{DD})	(excluding external current)						
	Active Mode		$I_{A/D}$				
	with A/D *)						
	T_A= 0 to 80 °C	5V			800	1100	µA
	T_A= -40 to 125 °C	5V			800	1400	µA
	Active Mode		I_{act}				
	without A/D *)						
	T_A= 0 to 80 °C	5V			400	500	µA
	T_A= -40 to 125 °C	5V			520	650	µA
	DONE Mode		I_{Done}				
	(Standby) *)						
	T_A= 0 to 80 °C	5V			10	18	µA
	T_A= -40 to 125 °C	5V			12	20	µA
	OFF Mode (Halt) *)		I_{OFF}				
	T_A= 0 to 80 °C	5V			0.1	1	µA
	T_A= -40 to 125 °C	5V			0.1	1	µA

*) Current values are for input levels in the range of
0...0.3 V For V_{KL}, V_{IOL};
(V_{DD} - 0.3 V) V_{DD} For V_{KH}, V_{IOH} - All outputs open

6.7 TSS405 -Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
K,I/Oinputs (Schmitt-Tr)							
V_{T+} (positive going threshold voltage)		5V	V_{KH}	2.7		3.9	V
V_{T-} (negative going threshold voltage)		5V	V_{KL}	1.0		1.9	V
V_{T+} - V_{T-} (Hysteresis)		5V		0.8		2.9	V
Option pull-down	V_{IN} = 0.6V	5V	I_{KR}	5	20	15	µA
	V_{IN} = 5V	5V	I_{KR}	50	160	125	µA
Option no pull-down	V_{IN} = 0V	5.5V	I_{KR}	-0.1		0.1	µA
	V_{IN} = V_{DD}	5.5V	I_{KR}	-0.1		0.1	µA
K as output "1"	I_{KHmax} = - 0.5 mA	5V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
Option push-pull	I_{KLmax} = + 0.4 mA	5V	V_{KL}	V_{SS}		V_{SS}+0.4	V
Option open-source	I_{KHmax} = - 0.5 mA	5V	V_{KH}	V_{DD} - 0.6		V_{DD}	V
	V_K = 3.8 V	5V	I_{KOL}	30		30	nA
I/O as output							
"1"	I_{IOHmax} = - 0.75 mA	5V	V_{IOH}	V_{DD} - 0.6		V_{DD}	V
"0"	I_{IOLmax} = + 1.0 mA	5V	V_{IOL}	V_{SS}		V_{SS}+0.4	V
R output							
"1"	I_{RHmax} = - 0.3mA	5V	V_{RH}	V_{DD} - 0.4		V_{DD}	V
"0"	I_{RLmax} = + 0.3 mA	5V	V_{RL}	V_{SS}		V_{SS}+0.4	V

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
Clock (Opt.:NONE,256 Hz, 1024 Hz) (not active at OFF mode)							
output "1"	I_{CH} = - 0.6 mA	5	V_{CH}	V_{DD} - 0.6		V_{DD}	V
output "0"	I_{CL} = + 0.5 mA	5	V_{CL}	V_{SS}		VSS + 0.4	V
INITN	V_{IN} = 0V	5	I_{INIT}	- 0.5		- 2.2	µA
	V_{IN} = V_{DD}	5.5	I_{INIT}	- 0.1		0.1	µA
SV_{DD} (switched V_{DD})	I_{SVDD} = 2 mA	2.6V	V_{SVDD}	V_{DD} - 0.2		V_{DD}	V
	I_{SVDD} = 6.5 mA	2.6V	V_{SVDD}	V_{DD} - 0.3		V_{DD}	
	SV_{DD} off (0V)	5.5	I_{SVDD}	- 0.1		0.1	µA
KC	V_{KC} = 0V	5.5	I_{KC}	- 0.1		0.1	µA
	V_{KC} = V_{DD}	5.5	I_{KC}	- 0.1		0.1	µA

6.7 **TSS405 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common (1/2 duty cycle)							
output "1"	I_{CH} = - 100 µA	5V	V_{CH}	V_{DD} - 0.4	$V_{DD}/2$	V_{DD}	V
"1/2 Hz"	I_{CHz} = +/-10 nA	5V	V_{Hz}	$V_{DD}/2$-.06		$V_{DD}/2$+.06	V
"1/2 Hz"	V_{CHz} = 0V	5V	I_{CHz}	-2.5		- 6.8	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	2.5		6.8	µA
output "0"	I_{CL} = + 100 µA	5V	V_{CL}	V_{SS}		V_{SS} + 0.4	V
LCD lines segment (1/2 duty cycle)							
output "1"	I_{CH} = - 100 µA	5V	V_{SH}	V_{DD} - 0.4		V_{DD}	V
output "0"	I_{CL} = + 100 µA	5V	V_{SL}	V_{SS}		V_{SS} + 0.4	V

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LCD lines common, segment (1/4 duty cycle)							
output "1"	I_{CH} = - 100 µA	5V	V_{CH}	V_{DD} - 0.4		V_{DD}	V
output "2/3 Hz"	I_{CHz} = +/- 10 nA	5V	V_{Hz}	2·V_{DD}/3-0.04	2·V_{DD}/3	2·V_{DD}/3+0.04	V
"2/3 Hz"	V_{CHz} = 0V	5V	I_{CHz}	- 2.5		- 6.8	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	1.25		3.4	µA
output "1/3 Hz"	I_{CHz} = +/- 10 nA	5V	V_{Hz}	V_{DD}/3-0.04	V_{DD}/3	V_{DD}/3+0.04	V
"1/3 Hz"	V_{CHz} = 0V	5V	I_{CHz}	- 1.25		- 3.4	µA
	V_{CHz} = V_{DD}	5V	I_{CHz}	2.5		6.8	µA
output "0"	I_{CL} = + 100 µA	5V	V_{CL}	V_{SS}		V_{SS} + 0.4	V

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext} = V_{SVDD} - V_{RI}							
Range large*	I_{RI} = 1.3 mA; T_A = 25 °C	5V	V_{Rext}	0.241959·SV_{DD}	0.244403·SV_{DD}	0.246847·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA,						
	AV_{LL}	5V	dN/dT		0.03		LSB/°C
	AV_{HL}	5V	dN/dT		0.06		LSB/°C
SV_{DD}	T_A = 25 °C,						
rejection ratio	AV_{LL}	5V	dN/ dSV_{DD}	- 7	- 1.5	1	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 6	- 3	2	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range large)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 02B_{16}$	5V	AV_{LL}	$0.103423 \cdot SV_{DD}$	$0.104345 \cdot SV_{DD}$	$0.107267 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = FD4_{16}$	5V	AV_{HL}	$0.488649 \cdot SV_{DD}$	$0.490571 SV_{DD}$	$0.492493 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FD4_{16}$ - $02B_{16}$)	5V	R_{AV} (AV_{HL} - AV_{LL})	$0.383784 \cdot SV_{DD}$	$0.385226 \cdot SV_{DD}$	$0.386667 \cdot SV_{DD}$	V

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		5V			$43.0 \cdot 10^{-6} \cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 **TSS405 - Operating Characteristics** (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	**Condition**	V_{DD}	**Symbol**	**Min**	**Nom**	**Max**	**Unit**
A/D Converter current source V_{Rext} = V_{SVDD} - V_{RI}							
Range small	I_{RI} = 1.3 mA; T_A = 25 °C	5V	V_{Rext}	0.237836·SV_{DD}	0.240238·SV_{DD}	0.242641·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} = 1.3 mA,						
	AV_{LL}	5V	dN/dT		0.07		LSB/°C
	AV_{HL}	5V	dN/dT		0.14		LSB/°C
SV_{DD}	T_A = 25 °C,						
rejection ratio	AV_{LL}	5V	dN/ dSV_{DD}	- 7	- 3.5	2.5	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 14	- 7	5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.7 TSS405 - Operating Characteristics (continued):

T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range small)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 0E5_{16}$	5V	AV_{LL}	$0.233596 \cdot SV_{DD}$	$0.235317 \cdot SV_{DD}$	$0.237039 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = FA1_{16}$	5V	AV_{HL}	$0.401786 \cdot SV_{DD}$	$0.403508 \cdot SV_{DD}$	$0.405230 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FA1_{16}$ - $05E_{16}$)	5V	R_{AV} (AV_{HL}- AV_{LL})	$0.167328 \cdot SV_{DD}$	$0.168189 \cdot SV_{DD}$	$0.169050 \cdot SV_{DD}$	V

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 **TSS405 - Operating Characteristics** (continued): $T_A = 0$ to 80°C, unless otherwise noted.

Parameter	**Condition**	V_{DD}	**Symbol**	**Min**	**Nom**	**Max**	**Unit**
LSB Voltage		5V			$43.0 \cdot 10^{-6} \cdot SV_{DD}$		V
Linearity	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source V_{Rext} = V_{SVDD} - V_{RI}							
Range medium-high*	I_{RI} = 1.3 mA ; T_A = 25 °C	5V	V_{Rext}	0.239187·SV_{DD}	0.241603·SV_{DD}	0.244019·SV_{DD}	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	5V	dN/dT		0.04		LSB/°C
	AV_{HL}	5V	dN/dT		0.08		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C,						
	AV_{LL}	5V	dN/ dSV_{DD}	- 4,5	- 2	1.5	LSB/V
	AV_{HL}	5V	dN/ dSV_{DD}	- 9	- 4,5	3	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-high)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 03E_{16}$	5V	AV_{LL}	$0.234711 \cdot SV_{DD}$	$0.236673 \cdot SV_{DD}$	$0.238635 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = FC1_{16}$	5V	AV_{HL}	$0.494451 \cdot SV_{DD}$	$0.496413 \cdot SV_{DD}$	$0.498375 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FC1_{16}$ - $03E_{16}$)	5V	R_{AV} (AV_{HL} - AV_{LL})	$0.258768 \cdot SV_{DD}$	$0.259750 \cdot SV_{DD}$	$0.260731 \cdot SV_{DD}$	V

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
LSB Voltage		5V			$65.4 \cdot 10^{-6} \cdot SV_{DD}$		
Linearity	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB

Note: AD-range limited due to offset of comparator. Nevertheless the range can be larger by smaller offset.

6.7 TSS405 - Operating Characteristics (continued): T_A = 0 to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter current source $V_{Rext} = V_{SVDD} - V_{RI}$							
Range medium-low	I_{RI} = 1.3 mA ; T_A = 25 °C	5V	V_{Rext}	$0.239841 \cdot SV_{DD}$	$0.242263 \cdot SV_{DD}$	$0.244686 \cdot SV_{DD}$	V
external Resistor A1, A2, A3, A4	T_A = 25 °C	5V	R_{ext}	0.10		1.6	kOhm
Temperature stability	V_{Rext}/R_{ext} =1.3 mA,						
	AV_{LL}	5V	dN/dT		0.04		LSB/°C
	AV_{HL}	5V	dN/dT		0.08		LSB/°C
SV_{DD} rejection ratio	T_A = 25 °C,						
	AV_{LL}	5V	dN/dSV_{DD}	- 4	- 2	1.5	LSB/V
	AV_{HL}	5V	dN/dSV_{DD}	- 8	- 4	2.5	LSB/V
Drift	V_{Rext}/R_{ext} = 1.3 mA	5V	dVR_{ext}/ month				

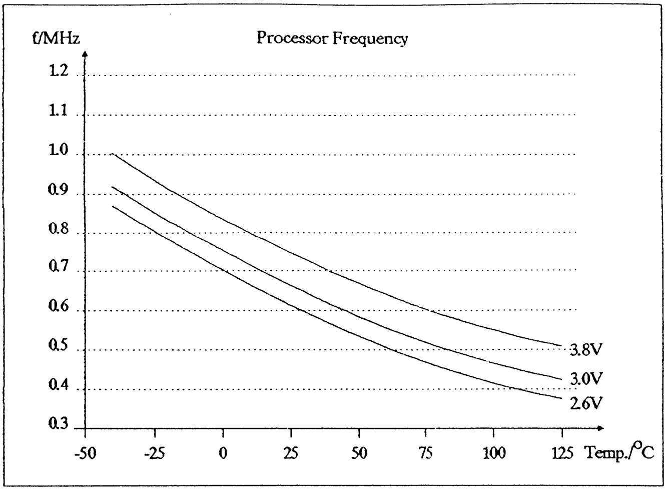
6.7 TSS405 - Operating Characteristics (continued): $T_A = 0$ to 80°C, unless otherwise noted.

Parameter	Condition	V_{DD}	Symbol	Min	Nom	Max	Unit
A/D Converter (Range medium-low)							
Input current (any inp.)	$V_{IN} = V_{SS}$ to V_{DD}; DL13 = 0		$I_{A1/2/3/4}$			+/- 30	nA
Conversion analog input to digital value							
Analog input voltage lower limit	$N_{LL} = 036_{16}$	5V	AV_{LL}	$0.102066 \cdot SV_{DD}$	$0.104281 \cdot SV_{DD}$	$0.106495 \cdot SV_{DD}$	V
Analog input voltage higher limit	$N_{HL} = FC7_{16}$	5V	AV_{HL}	$0.396227 \cdot SV_{DD}$	$0.359441 \cdot SV_{DD}$	$0.400656 \cdot SV_{DD}$	V
Range analog input volt. (see Note)	digital value ($FC7_{16}$ - 036_{16})	5V	R_{AV} (AV_{HL} - AV_{LL})	$0.293053 \cdot SV_{DD}$	$0.294161 \cdot SV_{DD}$	$0.295268 \cdot SV_{DD}$	V

6.7 **TSS405 - Operating Characteristics** (continued):
T_A = 0 to 80°C, unless otherwise noted.

Parameter	**Condition**	V_{DD}	**Symbol**	**Min**	**Nom**	**Max**	**Unit**
LSB Voltage		5V			$73.8 \cdot 10^{-6} \cdot SV_{DD}$		
Linearity	Delta digital value ≤ 120 LSB	5V		-1		+1	LSB
	120 LSB < DDV ≤ 240 LSB	5V		-1 1/2		+1 1/2	LSB
	240 LSB < DDV ≤ 2600 LSB	5V		-2 1/2		+2 1/2	LSB
	DDV >2600	5V		-4 1/2		+4 1/2	LSB
Battery check							
Range small, medium-high	000_{16}< conversion result >FFF_{16}		V_{DDBC}		2.7	3.7	V
Range large, medium-low	000_{16}< conversion result >FFF_{16}		V_{DDBC}		2.7	4.8	V
Drift	N of ADC @ V_{DD}=2.7V,25 °C			-0.1		+0.1	V

6.8 Typical RC-MOS Oscillator Characteristics



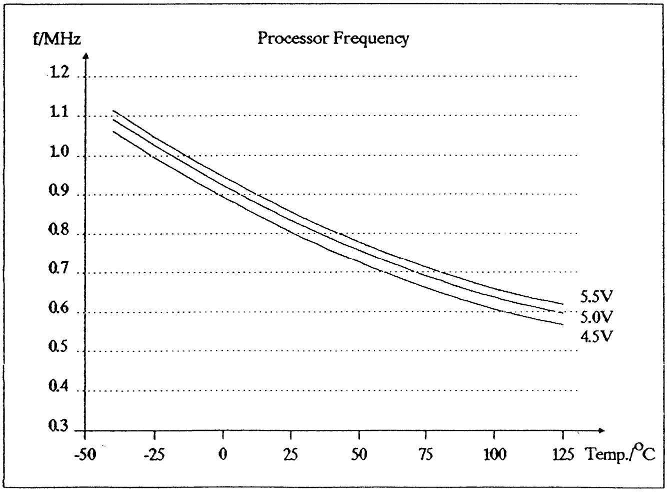


Figure 6-1:

6.9 Schematics of Inputs/Outputs

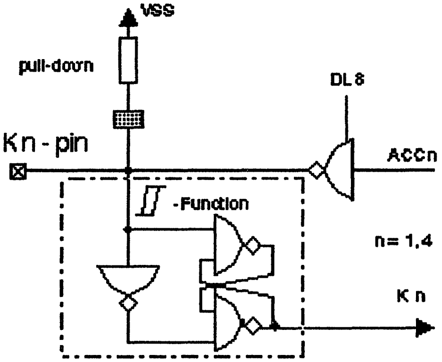


Figure 6-2: Schematic of K-I/Os

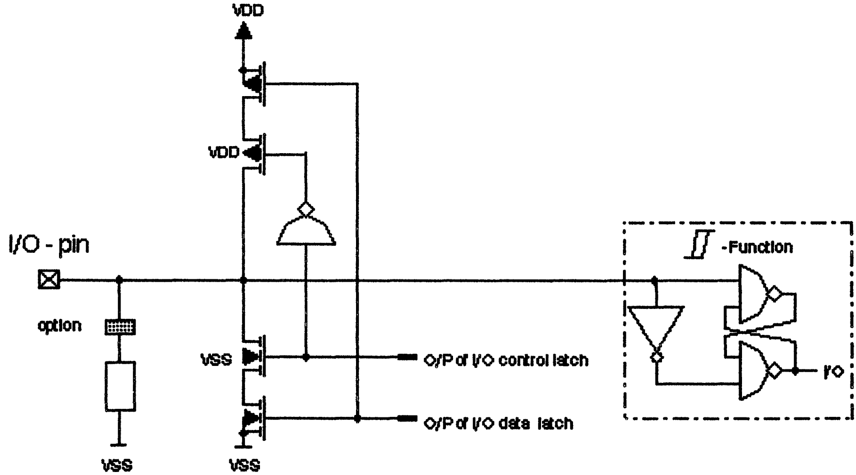


Figure 6-3: Schematic of I/O

6.9 Schematics of Inputs/Outputs (continued)

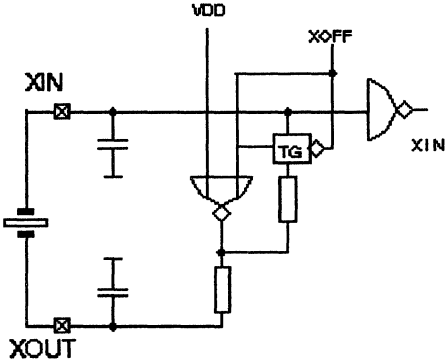


Figure 6-4: Schematic of XTAL Oscillator

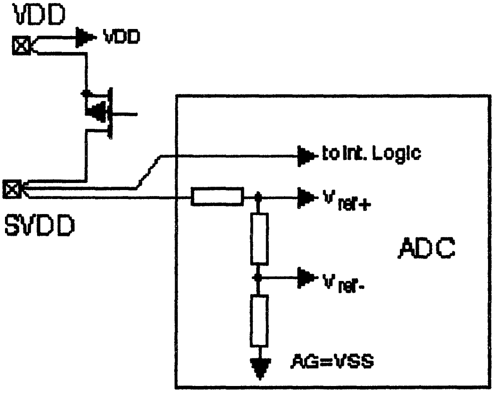


Figure 6-5: Schematic of SVDD

APPENDIX A - Software Restrictions

1. Carefully examine the states of inputs and outputs when DONE or OFF instruction is used. Prevent additional DC current in low power consumption mode caused by the states of inputs and outputs.
2. Pay attention to the states of I/O pin and the K-lines while being in DONE/OFF. The wake-up can be triggered by K1, K2, K4, K8 and I/O - but only if all were Low before one of these Lines goes High.
3. The processor begins execution at page 15; PC=00 the first time power is applied. Each time a "wake-up" signal occurs, the processor begins execution at page 15; PC = >7F.
4. For operations which require more then 15 seconds of processing, the software should periodically update the timekeeping since the chip provides only modulo-16-sec.
5. The SAL instruction may cause the End-Around-Carry (SS) to change state falsely in either direction. Thus, after an SAL, the SS should be set to the desired state with a SEAC or REAC. Also, if both the Add latch and SS must be set to a desired state, the SAL should be done first. For example, in the basic Add loop where both the Add latch must be set and the SS reset, the SAL should be executed prior to the REAC.
6. The TTA and TTM instructions:
   It is possible to get a false intermediate reading because of the Timer and Processor operate asynchronous. To prevent this, the time check must be repeated until two successive tests yield the same value.
7. Between a wake-up from the timer and a DONE instruction there must be a minimum number of instructions N:

$$N >= \frac{4 \text{ x processor frequency}}{6 \text{ x } f_{TOSCIN}} + 1$$

8. To execute a three instruction BCD addition, the following sequence of instructions may be used:

LOC		**INSTRUCTION**		**COMMENT**
001		SBL		Before jumping to subroutine, set Branch Latch
002		CALL	START	
"		"		
"		"		
"		"		
010	ADDBCD	TAMACS	6	
011		CTMDYN		
012	START	BRANCH	ADDBCD	Branch to location 10 and perform a DMEA instruction (code 10) before executing TAMACS
013		CCLA		
014		RETN		

9. SBL instruction:
   If Branch Latch is set use only Branches which can operate on an instruction. In other ways it is possible that some of the micro-instructions are active and will cause some problems (i.e., transfer CKB-Bus to memory....). Don't forget to reset Branch Latch after using it.

10. Processor enable logic "wake-up"
    Between TKA and DONE/OFF instruction not more than three instructions are allowed, if a successful "wake-up" should be performed by external hardware (→ K inputs / I/O).

11. Processor logic "wake-up" enable
    If the option of "wake-up" the processor by the negative and/or positive edge of K8 is used, pay attention to the effect: inverting K8 by DL14 can produce an edge. If this happens less than 7 instructions before the DONE instruction the processor could "wake up" due to the integrated delay line.

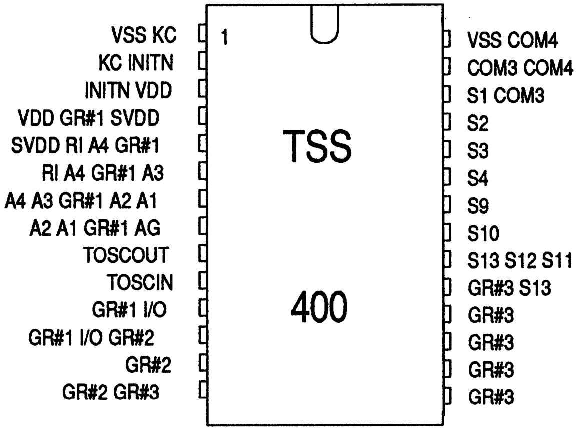
12. Table Status after "wake-up"

Register content after entry into active Processor-Mode

Register		Power up	INITN pin	DONE Mode	OFF Mode
Program Counter	PC	00	00	7F	7F
Page Address	PA	F	F	F	F
Page Buffer	PB	F	F	F	F
Status	S	undefined	undefined	undefined	undefined
Special Status	SS	undefined	undefined	undefined	undefined
Accumulator	A	undefined	undefined	undefined	undefined
YRegister	Y	undefined	undefined	undefined	undefined
XRegister	X	undefined	undefined	undefined	undefined
RAM Content		undefined	unchanged	unchanged	unchanged
DAM Latch		undefined	undefined	undefined	undefined
ADD Latch		reset	reset	reset	reset
BRANCH Latch		reset	reset	reset	reset
SEG H Decimal point	DP	reset	reset	reset	reset
Digit Latches	DLn	reset	reset	unchanged	unchanged
K-Port Data Latch		undefined	unchanged	unchanged	unchanged
I/O Port Data Latch		undefined	unchanged	unchanged	unchanged
I/O Port Control Latch		reset	reset	unchanged	unchanged
Timer0 and Timer1		undefined	unchanged	actual Time	undefined
ADC/Pin Supply SVDD		switched off	switched off	switched off	switched off
LCD Segment Latches		undefined	unchanged	unchanged	unchanged

APPENDIX B - Pinning Selection Schematic

28 PIN DUAL-IN-LINE PLASTIC PACKAGE



Note:

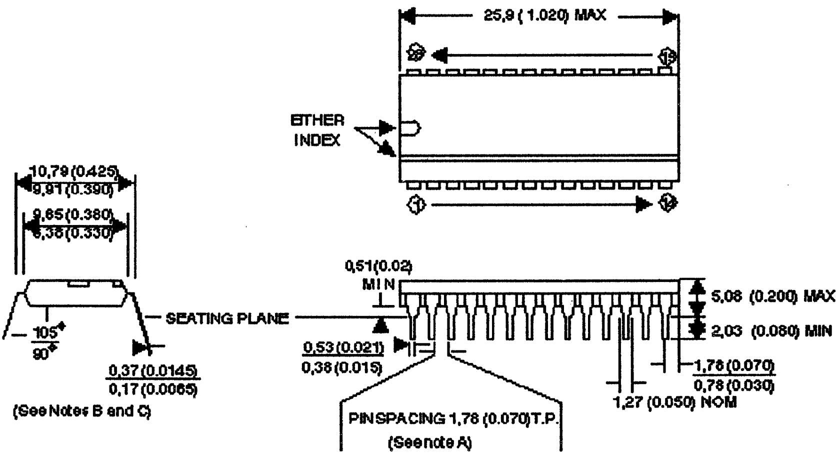
mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

APPENDIX B - Pinning Selection Schematic (continued)

28-pin NF package (YS)



All dimensions are in millimeter and parenthetically in inches.

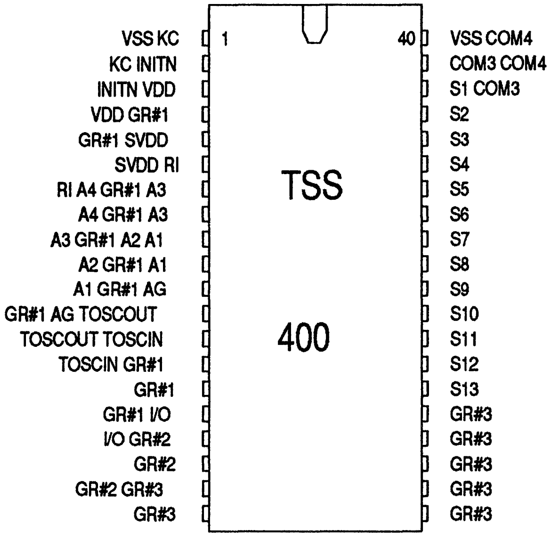
Notes: A: Each pin centerline is located within 0,25 (0.010) of its true longitudinal position.

B: This dimension does not apply for solder-dipped leads.

C: When solder-dipped leads are specified, dipped area of the lead extends from the lead tip to at least 0,51 (0.020) above seating plane.

APPENDIX B - Pinning Selection Schematic (continued)

40 PIN DUAL-IN-LINE PLASTIC PACKAGE



Note:

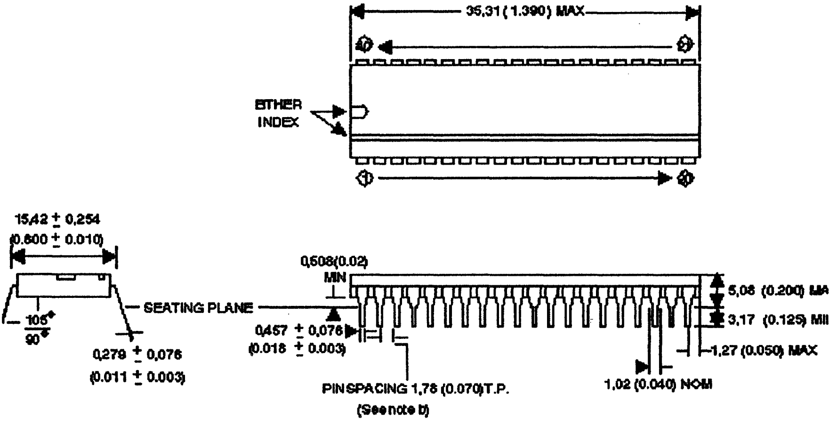
mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

APPENDIX B - Pinning Selection Schematic (continued)

40-pin N2 package (YS)

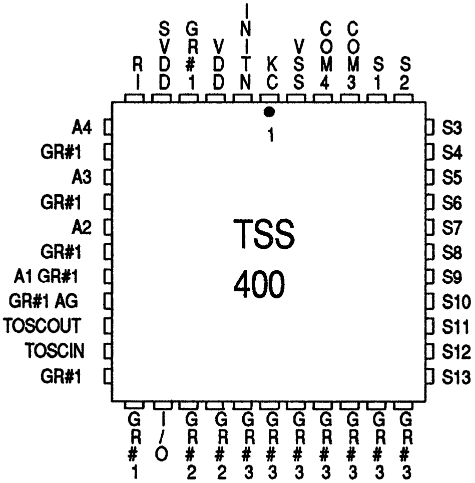


Note: A: All dimensions are in millimeter and parenthetically in inches.

B: Each pin centerline is located within 0,26 (0.010) of it's true longitudinal position.

APPENDIX B - Pinning Selection Schematic (continued)

44 PIN PLASTIC LEADED CHIP CARRIER PLCC



Note:

mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

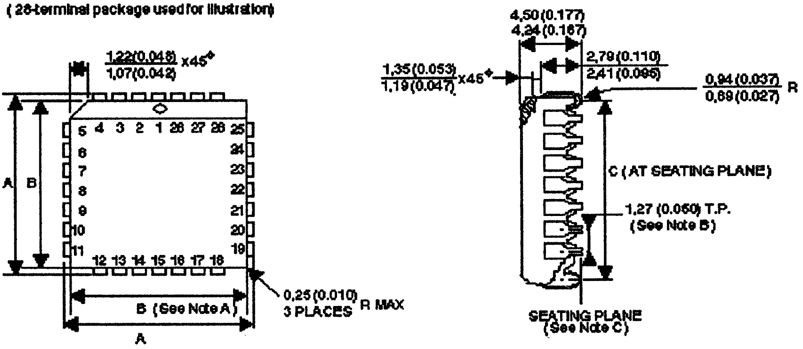
GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

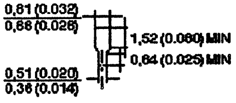
APPENDIX B - Pinning Selection Schematic (continued)

FN Plastic Chip Carrier Package

(28-terminal package used for illustration)



JEDEC Outline	No. of terminals	A min.	A max.	B min.	B max.	C min.	C max.
MO-047AC	44	17,40 (0.685)	17,65 (0.695)	16,51 (0.650)	16,66 (0.656)	15,49 (0.610)	16,00 (0.630)



LEAD DETAIL

Notes: A: Centerline of center pin each side is within 0,10 (0.004) of package centerline as determine by dimension B.

B: Location of each pin is within 0,127 (0.006) of true position with respect to center pin on each side.

C: The lead contact points are planar within 0,10 (0.004).

All dimensions and notes for the specified JEDEC outline apply.

All linear dimensions are in millimeters and parenthetically in inches.

APPENDIX C - User Options

LCD

- Segment PLA with 64 terms:
  4bit ACCU + Status + DL14 define on/off of segments A to G. Segment H is defined by instructions SDP and RDP.
- Common lines
  2MUX (1/2 duty) or 4MUX (1/4 duty). (Some manufacturer use back plane instead of common).
- Segment lines
  1. Attachment segment to common.
  2. Attachment segment to Y-Register.

Note: All three options must be defined even if no display will be used.

WAKE-UP FREQUENCY

- DL12 = 0 : 1 Hz (no option)
- DL12 = 1 : 128 OR 64 OR 32 OR 16 OR 8 Hz only one must be selected.

PROCESSOR FREQUENCY

- Internal RC-MOS Oscillator (see Figure 6-1).
- External Supply on Pin PFRQ (see 6.3 Operating Conditions)

Note: By using the "EXTERN" Option the internal Osc can be overwritten on Pin "PFRQ".
External Signal strength required >10 µA @ V_{DD}-0,4 V, V_{SS}+0,4 V

ADC RANGE

- SMALL
- MEDIUM LOW
- MEDIUM HIGH
- LARGE

Note: One and only one must be selected.

TIMOSC

- TRC	RC Oscillator
- TXOSC	Christal Oscillator

If the RC Oscillator is selected, a resistor needs to be connected from TOSCIN to TOSCOUT and a capacitor from TOSCIN to VSS. Please make sure that the frame frequency is o.k. if an LCD is used.

PULL-DOWN

Individually selectable for K1, K2, K4, K8 AND I/O.

Kn OPEN-SOURCE (n = 1,2,4,8)

Individually selectable for K1, K2, K4 and K8 to be open source

CLOCK

Frequency output divided from the crystal oscillator (XCOSIN).

- NONE	: nothing selected - no (bond)pad used
- 256 Hz	: $f_{CLOCK} = f_{TOSCIN}/128$
- 1024 Hz	: $f_{CLOCK} = f_{TOSCIN}/32$

K8 FLIP-FLOP

Signal on K8 line not immediate available for ACCU/MEMORY; if this option is used, signal at K8 pin is divided by 2 and then available for further operations (TKA, TKM, KNEZ). The divider by 2 is reset by PUC (power-up clear) or INIT.

PINNING

Variable pinning helps optimization of system cost including PCB.

TSS400 Family User's Guide

Sensor Signal Processors

Sensor Signal Processors

Part II

TSS400 Family User's Guide

Sensor Signal Processors

TSS400/4 USER'S GUIDE
October 1991

Table of Contents

Purpose and Conventions

LIST of FIGURES

Purpose and Conventions

The TSS400/4 User's Guide is intended to aid the development of TSS400 products by collecting and presenting hardware and software information in a manner that will be quickly accessible to engineers and programmers.

The following substitutes have been adopted because typographical limitations do not allow the use of some conventional symbols:

Symbol	Meaning	Example
<>	"Not equal"	Y <> A
[]	Subscript number	R[0]
^	Exponent	T^2
µsec	Microsecond	
msec	Millisecond	
>XX	Hex-Value	>3F4

1 TSS400/4

This document describes in detail the functional and electrical characteristics of a 4-bit CMOS sensor signal processor with computing capability. Its key applications are A/D conversions, calculating, controlling and displaying information on an LCD display. The part easily interfaces with sensors and actuators.

Typical applications are:
- temperature measurements: calculating, controlling, warning
- pressure and acceleration measurements
- home appliances
- intelligent keyboard and display driver
- timer with control functions
- intelligent subsystem

The TSS400/4 is an extension of the TSS400. It holds 4K instead of 2K ROM and contains 960 instead of 576 bits of RAM.

1.1 Functional Specification

- 4096 ten bit words of ROM
- 960 bit static RAM
- Three instruction BCD addition
- 64 character, 7 segment output PLA (DP switchable independently)
- Three levels of subroutine
- Timekeeping capability (32 768 Hz XTAL)
- Direct drive of LCD: (32 768 Hz XTAL)
  Option 2MUX: 1/2 duty cycle containing up to 40 segments
  Option 4MUX: 1/4 duty cycle containing up to 80 segments
- Low power silicon gate CMOS process:
  - Low power consumption at sleep mode (active timer/RAM)
  - very low power consumption at off mode (active RAM)
- 12 Bit A/D-Converter with 4-MUX-inputs and programmable ranges
- Programmable current source from 0.15 to 2.4 mA x VDD/V
- Internal MOS oscillator (see Figure 6-1)
- Instruction execution time: 5.5 µs (@f_{proc} = 1.1 MHz)
  15.0 µs (@f_{proc} = 400 KHz)
- Processor frequency determined either by the internal RC-MOS oscillator or the external clock input

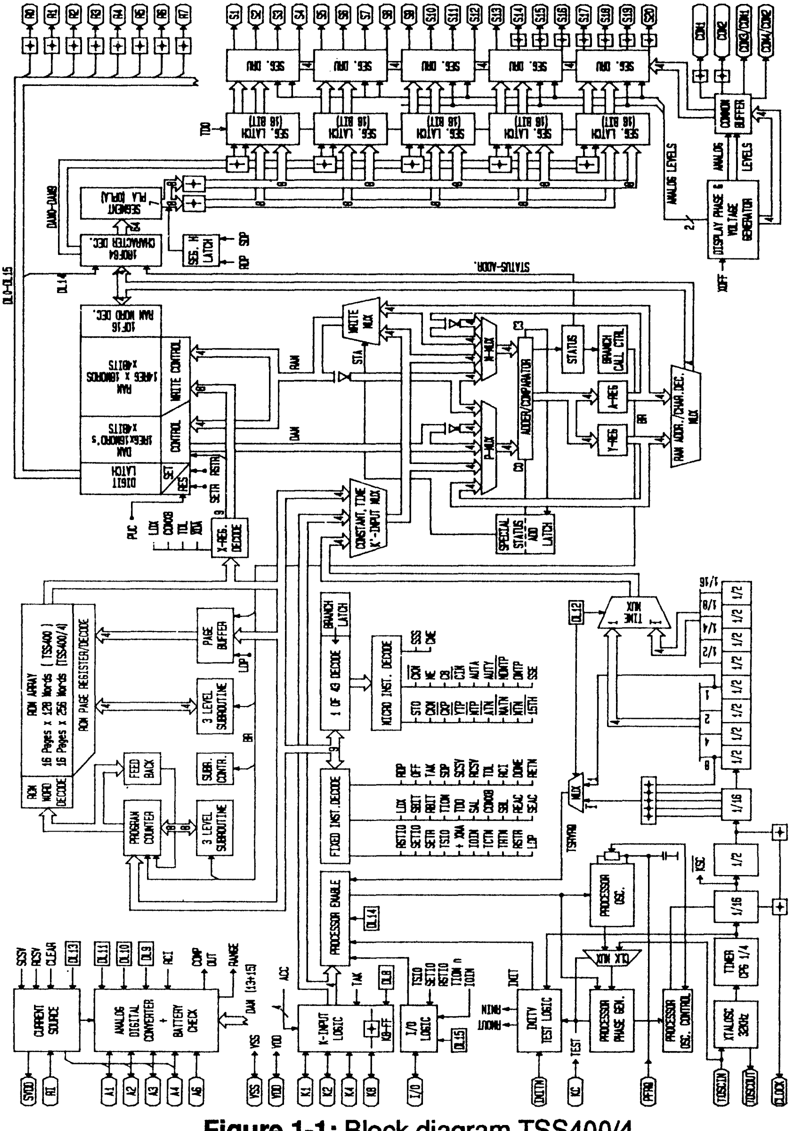


Figure 1-1: Block diagram TSS400/4

1.2 Memory Structure

The TSS400/4 has fifteen (15) registers with sixteen (16) four-bit words per register.

Fourteen (14) registers are random access memories (RAM) and one (1) register is a direct access memory (DAM).

The 15 registers can be addressed by the X-REG. The content of the X-REG (4 bits) is controlled by the software instructions LDX, TDL, COMX8.

The Y-Register (4 bits) addresses the word within the selected Register. There are several different software instructions that control the contents of the Y-REG.

Since the TSS400/4 has 15 registers, two values of the X-REG address the same memory. Figure 1-2 is a list of which values will address memory.

X-REG	HEX	TYPE
0000	0	RAM
0001	1	RAM
0010	2	RAM
0011	3	RAM
0100	4	RAM
0101	5	RAM
0110	6	RAM
1000	8	RAM
1001	9	RAM
1010	A	RAM
1011	B	RAM
1100	C	RAM
1101	D	RAM
1110	E	RAM
0111	7	DAM*
1111	F	DAM*

* Both >7 and >F address the same register (the DAM).

Figure 1-2: X-REG Values which Address Memory

In addition to the memory controlled by the X-REG and the Y-REG there are sixteen (16) digit latches which can be set and reset by software instructions. These digit latches are reset by hardware during power-up or by $\overline{\text{INIT}}$. The digit latches are addressed by Y-REG (4 bits) values >0 - >F.

1.3 Subroutine Stack Operation

The Subroutine Stack has the following registers:
1) Page ROM Address Stack Register (3)
2) Program Counter Stack Registers (3)

The interaction of Page Register, Page ROM Address and Stack is shown for the execution of subroutines.

P-Reg	X	X	2	2	2	3	3	4	4	5	5	6	6	6	6	6	6	6	6	6	6
ROM Addr. Reg.	15	15	15	15	15	15	3	3	4	4	5	5	6		4	4	4	4	3	3	3
Sub-routine Reg. A	X	15	15	X	X	X	X	X	3	3	4	4	5	4	3	3	4	3	15	15	15
Sub-routine Reg. B	X	X	X	X	X	X	X	X	X	X	3	3	4	3	15	15	3	15	15	15	15
Sub-routine Reg. C	X	X	X	X	X	X	X	X	X	X	X	X	3	15	15	15	15	15	15	15	15
	PUC	CALL	LDP2	RETN	BR	LDP3	BR*	LDP4	CALL	LDP5	CALL	LDP6	CALL	RETN	RETN	BR	CALL	RETN	RETN	RETN	RETN

* This branch is executed (branch occurs).

Figure 1-3: Subroutine Stack Operation (See Section 2.2 and 2.3 for more details.)

1.4 Digit Latches/R Outputs

Digit Latch	**Hardware Function**
DL0-DL7	Stat. outputs R0-R7 → max. 8 R outputs available
DL8	Control of K-Port: DL8 = 0 -- Input DL8 = 1 -- Output
DL9-DL11	Analog Mux for ADC: A1-A2-A3-A4 or Battery check
DL12	Selection of 1 Hz (DL12=0) or selection of mask programmed frequency for wake-up of the processor. Also selection of time info to CKB-BUS.
DL13	DL13 — Function 0 — constant current off 1 — constant current on
DL14	MSB of character decode and selection of K8 wake-up edge: leading (DL14 = 0) or trailing (DL14 = 1)
DL15	DL15 — Function 0 — wake-up of I/O enabled 1 — wake-up of I/O disabled

Figure 1-4: Digit Latches/R Outputs

PUC-signal will reset all Digit Latches.

The digit latches are set by the SETR instruction and are reset by the RSTR instruction.

The value of the Y-REG determines which digit latch is affected. The data of the Digit Latches will be retained when the product is in Sleep mode or Off mode as long as the power is maintained.

1.5 DAM Operation

The DAM outputs are transferred into the ALU input regardless the values of the X-REG. This allows operation on data in DAM and one of the 14 memory banks (RAM) at the same time. The DMEA instruction is an example.

To write into the DAM, the content of the X-REG has to be 0111 or 1111. This can be accomplished by a LDX 7, LDX 15, or a TDL instruction.

If DAM is addressed by the X-REG, all instructions operating on memory will operate on DAM. An example of this is the execution of a TMA instruction while the X-REG contains a 7. This causes the transfer of the DAM into the ACC.

If X-register is 0111 or 1111 instructions which use RAM and DAM will use DAM only. An example of this is the DMEA instr.:

DAM + MEM (= DAM) + SS → ACC   DAM*2 + SS → ACC

1.6 ALU

The ALU (Arithmetic Logic Unit) can perform addition, subtraction (by two's complement) and comparison of 2 four bit numbers. Data from the Y-register, DAM, CKB multiplexer or RAM is selected by the P-multiplexer and added to or compared with data from CKB, Accumulator or RAM selected by the N-multiplexer. The adder-comparator portion of the ALU adds or compares the data. The status will be set or reset depending on the result. Data from the adder can be transferred to the Accumulator, the Y-register, or can be dumped in case the data is needed for comparison only.

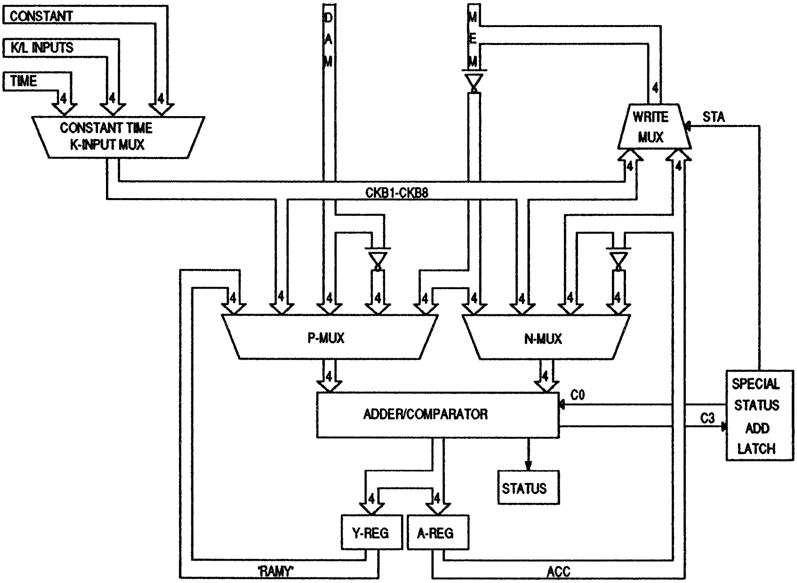


Figure 1-5: Block diagram of ALU

1.7 CKB Memory Map Description

The TSS400/4 has a 4-bit wide bus line (CKB) that may carry constant data, K-line data, timekeeping data or bit data (SBIT, RBIT, TBIT, TBITA, TCTM, TIOM). Any of these instructions determine which data are valid on the bus line at that time. In order to minimize the physical size of the CKB logic, certain portions of the instruction map are reserved by the CKB logic for data from one (1) of the four (4) sources to be on the CKB bus line.

Instruction Map Locations	Source of Data on CKB Bus
008 → 00F	K-lines
018 → 01F	TKP lines (timekeeping)
020 → 02F	Bit data for TBIT, TCTM
0A0 → 0AF	Bit data for SBIT,RBIT,TBITA,TIOM
040 → 07F	Constant data (I5 → I8)
0C0 → 0FF	Constant data (I5 → I8)

Figure 1-6: Instruction Restrictions

As shown in Figure 1-6, certain areas of the instruction map are decoded by the CKB logic and the CKB bus line will use data from one of its sources. Use micro-instructions CKM, $\overline{\text{CKP}}$, or $\overline{\text{CKN}}$ to transfer data to another portion of the processor. Please note that the source of data on the CKB bus is solely determined by a location in the instruction map as shown above.

1.8 K-Line Structure

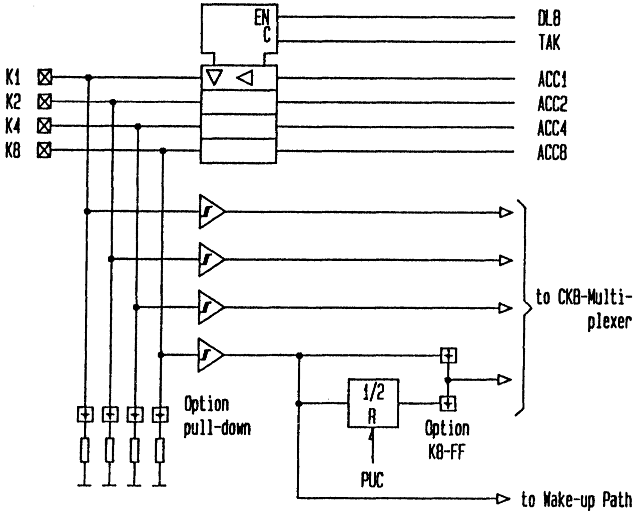


Figure 1-7: K-Line Structure

The present state of Digit latch DL8 defines the state of K-Lines to be Input or Output:

Input if DL8 = "0"
Output if DL8 = "1".

PUC signal will reset DL8 to "0" - Input.

The TAK instruction latches the 4 bit ACC into the output data register. If DL8 is "1" the data is available at the K-Lines:

K1 --- ACC1
K2 --- ACC2
K4 --- ACC4
K8 --- ACC8

The signals, applied at the K-pins, are read or tested directly or divided:

Option K8-FF not used	**Option K8-FF used**
K1 → ACC1, MEM0	K1 → ACC1, MEM0
K2 → ACC2, MEM1	K2 → ACC2, MEM1
K4 → ACC4, MEM2	K4 → ACC4, MEM2
K8 → ACC8, MEM3	K8/2→ ACC8, MEM3

Three instructions are implemented to read or test the information, applied at the K-Lines:

TKA, TKM and KNEZ.

1.9 I/O Structure

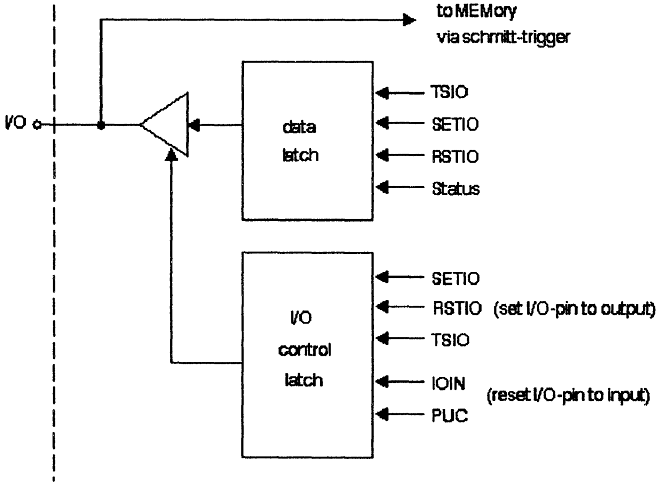


Figure 1-8: I/O Structure

The PUC signal as well as the IOIN instruction will reset the I/O pin to be an input (output buffer is 3-state).

Three instructions define the state of the I/O pin to be an output and will determine the data to be output:
- SETIO : set I/O pin to "1"
- RSTIO : reset I/O pin to "0"
- TSIO : output Status from the previous instruction output.

1.10 Clock

The Clock Output has 3 options that can be selected:

NONE : not selected

256 Hz : $f_{Clock} = \frac{f_{TOSCIN}}{128}$

1024 Hz : $f_{Clock} = \frac{f_{TOSCIN}}{32}$

When the processor is forced into the OFF mode, the oscillator is stopped and the TOSCIN pin is forced to V_{DD} via a p-channel transistor. The signal at the pin 'Clock' remains at the logical level it had during executing the OFF instruction.

When using the Clock Signal together with the OFF mode pay attention to the application S/W and H/W.

If a defined logical level is necessary in the application - e.g. to avoid current consumption - the logical level has to be read e.g. by K-input to check the state of the Clock Signal before executing the OFF instruction.

Note: The signal at the pin 'Clock' is not the same as the signal at the Pin "TOSCIN".

2 Operational Description

There are three different machine states: DONE, ACTIVE and OFF. During the DONE state only display and timekeeping circuitry are active. DONE is a low power state.

In the ACTIVE state the chip is executing instructions from its internal ROM and maximum power is dissipated.

In the OFF state only RAM/DAM and the actual level of I/O, R outputs and K-I/O's remain unchanged. This is true in the DONE state as well.

The machine state transitions are illustrated in Figure 1-12.

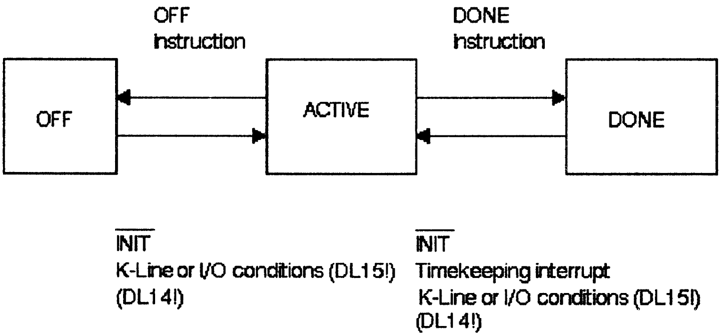


Figure 2-1: Machine Status

2.1 ROM Decode

The ROM on the TSS400/4 has sixteen (16) pages with 256 words per page and ten bits per word. The ROM page address is contained in the ROM Page Address Register. The ROM word address is derived from the program counter.

The four bits of the page register are decoded in a 1 of 16 decode. A bus runs from the 1 of 16 page decode to the ROM and addresses the proper page.

The eight bits of the program counter are decoded in a 1 of 256 decode. A bus runs from the 1 of 256 word decode to the ROM and addresses one word on each page. Page and word decode select together the particular word.

2.2 Program Counter and Subroutine Register

A bus from the program counter (PC) runs to a 1 of 256 decode and to a feedback network, which is used to change the state of the program counter in order to obtain the next ROM word address. When a CALL instruction is executed, the eight PC bits are stored in the first of a three levels of the subroutine stack. Each successful CALL instruction pushes down the stack. As many CALL instructions as necessary may be executed, but only the three most recent return addresses are stored. All previous return addresses are lost. Each RETN instruction, up to a maximum of three consecutive, pops up the stack. Any RETN instructions encountered when the device is not in a subroutine will be treated as no-op instructions.

2.3 ROM Page Address Register and Page Buffer

The ROM page address register, like the program counter, is capable of three levels of subroutines.

When a CALL instruction is executed, the four page-address bits are stored in the first of a three level subroutine stack. Each successful CALL instruction pushes down the stack. As many CALL instructions as necessary may be executed, but only the three most recent return addresses are stored. All previous addresses are lost. Each RETN instruction, up to a maximum of three consecutive, pops up the stack. Any RETN instruction encountered when the device is not in a subroutine will be treated as a no-op instruction.

When a LDP instruction is executed, the page address is loaded into the Page Buffer. The Page Address is transferred to the ROM Page Address Register when a CALL or BRANCH instruction is successfully executed. If the CALL or BRANCH instruction is unsuccessful or an RETN instruction is executed between the LDP and CALL instructions, the Page Buffer and the ROM Page Address Register remain unchanged.

After an LDP-RETN sequence, the ROM page Address Register remains unchained by CALL and BRANCH instructions.

2.4 Instruction Decode

First a ten-bit word from the ROM goes to the instruction decode, where three types of instructions are decoded:
- programmable instructions
- fixed instructions
- CKB instructions

If 1 of the 48 programmable instructions is decoded as true, a set of micro-instructions will be decoded. These micro-instructions control lines which handle one space too much various logic blocks.

If 1 of the 29 fixed instructions is decoded as true, the output of the respective instruction will set a control line used by a particular logic circuit.

If a CKB instruction is decoded as true, the 4 least significant bits of the ROM word, or the 4 K-lines, or the 4 timekeeping lines are passed into the ALU (depending on which instruction is decoded).

2.5 Processor Enable and Time Base Multiplexer

The software determines when the processor is disabled by executing a DONE instruction. The hardware keeps a timer up to 15 seconds; the software must keep time for all times part 15 seconds (minutes, hours, a.m/p.m., day of the week, date, and month). A calculation must not take longer than 15 seconds. In case a calculation does take longer by software an periodical interrupt of the calculation must occure to enable the time keeping update.

The processor enable logic "wakes-up" the processor from DONE state depending on DL12, to allow the update of the timekeeping.

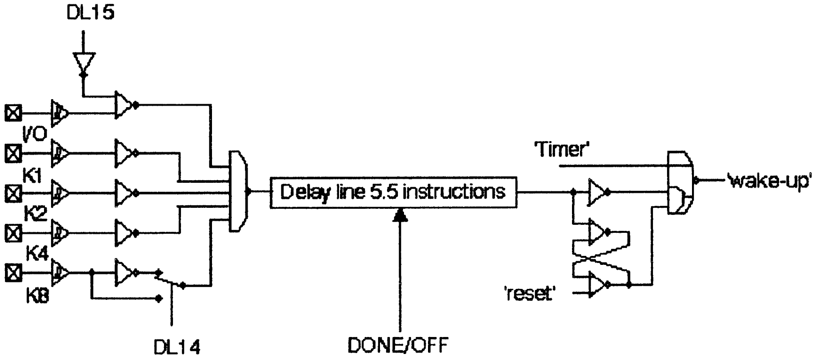
The processor-logic-enable also senses the K-lines K1, K2, K4, K8 and I/O line.

"Wake-up" from OFF state is only possible via Init, K-lines or I/O line; program execution starts at page 15, PC = >FF.

DL14	DL15	K1		K2		K4		K8		I/O	conditions for external wake-up
0	0	0	^	0	^	0	^	0	^	0	before wake-up will be accepted
0	0	↑	v	↑	v	↑	v	↑	v	↑	transition to wake-up processor
1	0	0	^	0	^	0	^	1	^	0	before wake-up will be accepted
1	0	↑	v	↑	v	↑	v	↓	v	↑	transition to wake-up processor
0	1	0	^	0	^	0	^	0	^	X	before wake-up will be accepted
0	1	↑	v	↑	v	↑	v	↑	v	X	transition to wake-up processor
1	1	0	^	0	^	0	^	1	^	X	before wake-up will be accepted
1	1	↑	v	↑	v	↑	v	↓	v	X	transition to wake-up processor

Note: 'v' - logical OR, '^' - logical AND, X - don't care

Figure 2-2: Table for ext. Wake-up Conditions



Note: Wake-up occurs when 'wake-up' signal is Low AND processor in DONE/OFF mode
Delay line is active if processor is active (5 to 6 instructions)
Delay line is inactive if processor is in DONE/OFF mode: No Delay
DL14 selects the logical state of K8 for 'wake-up'signal

Figure 2-3: Schematic of Wake-up Path

2.6 Processor Frequency

The processor frequency is derived from the internal RC-MOS Oscillator. The typical characteristic of this Oscillator is shown in Figure 6-1.

This internal MOS Oscillator frequency can be overwritten by an external Oscillator connected to Pin PFRQ. The frequency range for the external Oscillator is defined in Chapter 6.2 Operating Conditions.

2.7 Timer Oscillator

Crystal Oscillator

A 32.768 Hz signal of the crystal oscillator is fed into the crystal clock generator, which divides the frequency by four and generates four phase clocks. These clocks are divided down through a series of divide-by-two circuits into the frequencies needed for the timekeeping functions. A signal KSC is produced by the dividing circuit. This signal is used in by logic of the Common Generator, to generate the signals on the two or four common segment lines.

RC Oscillator

Using the RC oscillator (OPTION "TRC") instead of the crystal oscillator, the same applies as described in Section 2.7 (see Figure 2-4).

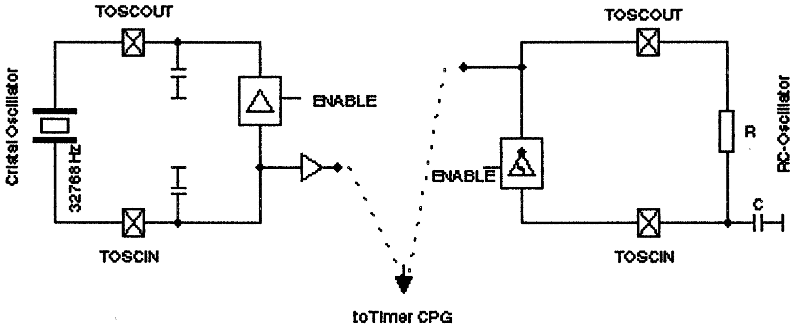


Figure 2-4: Option TIMOSC

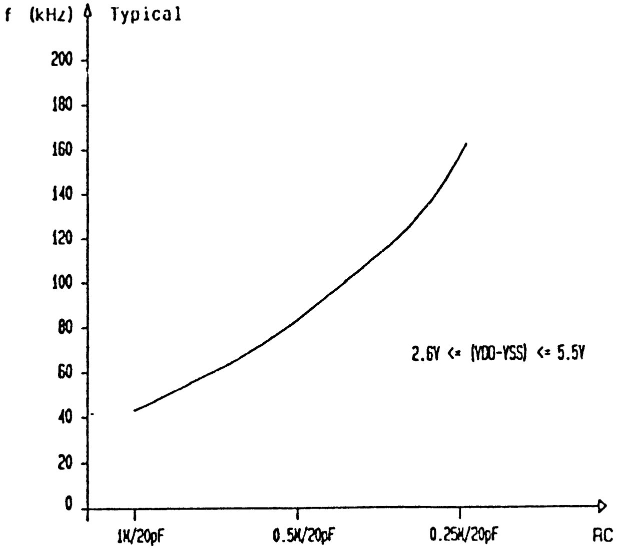


Figure 2-5: TRC Oscillator Frequency

2.8 Time Data Multiplexer

The TSS400/4 processor does not need to be active all the time. The processor is turned on only when needed to perform a software function or to update timekeeping. The processor can "wake-up" to update timekeeping either once per second or x-times per second (Option).

This is determined in the Time Data Multiplexer. When DL12 is high, the processor wakes-up x*-times per second, and when DL12 is low, the processor wakes-up once per second. This also determines the data used timekeeping.

* Options for x: 128 or 64 or 32 or 16 or 8

When DL12 is high, the 4-bit timekeeping data represents a time of 0/16 to 15/16 of a second, and when DL12 is Low, the data represent a time of 0 to 15 seconds.

The timekeeping can be accessed via CKB-Bus by software.

2.9 CKB (Constant, K-lines, Bit and Time Multiplexer)

All constant data used by the Y-REG, the Accumulator, or memory must pass the CKB. The logic of the CKB determines according to the used instruction whether to select constant, K-lines, bit logic, or timekeeping data to be transferred to various parts of the TSS400/4.

2.10 Branch/Call Control

When a BRANCH is encountered the Branch/Call Control Logic effects the program counter to dump the present ROM word address and accept a new one. When a CALL is encountered the Branch/Call Logic effects the program counter and the ROM page register to move their present contents into the subroutine latches and accept new addresses. A RETN forces the Branch/Call Control Logic to return the program counter to the stored address plus one. The stored address is the address when the call was encountered. The ROM page address register returns to the address it contained before the CALL. If a RETN is encountered while the TSS400 is not in a subroutine, it is regarded as a no-op and ignored. (See Status paragraph).

2.11 RAM Address/Character Decode Multiplexer

The RAM Address/Character Decode Multiplexer combines 2 blocks:

1) Multiplexer between the Y-Reg and the Accumulator. The output of the multiplexer is fed into the Decoder.
2) Decoder which decodes either the RAM word address from the Y-Register or the LCD character address from the Accumulator.

2.12 Write Multiplexer

The Write Multiplexer determines whether data is to be put into the RAM, the DAM, or the P-Multiplexer in the ALU. The data is taken from the CKB logic or from the Accumulator. The Write Multiplexer is controlled by software.

2.13 X-Decode

The X-REG is decoded by the X-Decode to select a particular RAM bank. The value of the DAM latch, which enables DAM and disables RAM or vice versa is also decoded by the X-REG.

2.14 RAM Word Decode

The RAM Address/Character Decode Multiplexer feeds all 4 bits of the Y-REG into a 1 of 16 decode. The state of the sixteen outputs of the RAM word decode which of the sixteen words in each bank of the RAM and DAM or which of the sixteen Digit latches is addressed.

2.15 Character Decode

The Character Decode uses DL14, status, and the four Accumulator bits to do a 1 of 64 character decode. The 64 signals are then transferred into the output PLA, where they are converted into a seven-segment output. (The Character Decode is illustrated in the Gate Level PLA paragraph.)

2.16 Segment Latches

After decoding and converting the characters into segment outputs, they are latched in Segment latches when the proper Y-pointer is set and a TDO instruction is executed.

2.17 Segment Drivers

The segment data is transferred to the Multiplexing Segment Drivers which convert the seven segments and the Segment H information into 2Mux or 4Mux segment information, which is output and used to turn on the LCD segments.

In test mode some of the LCD-Outputs do have other functions.

2.18 Power Up Clear

There is an time interval between power is applied to the TSS400/4 and the starting of the clocks. The Power Up Clear (PUC) signal sets the page and word address and enables the PC after the clocks have started.

When the voltage is applied, $\overline{\text{PUC}}$ starts the program execution on page 15, PC = >00. A low signal at $\overline{\text{INIT}}$ pin creates the same function as power being applied.

When the Processor is activated from Done or OFF mode via a K-input-I/O or by a timekeeping update, the program execution begins at Page 15, PC = >FF.

2.19 Status

Status is held in a single bit latch. When this bit is set, which is its normal state, and a BR or CALL is encountered, the BR or CALL will be executed. When status is reset immediately before a CALL or BR, the instruction will not be executed successfully and the Program Counter will continue incrementing.

Status will remain reset (0) for only one instruction cycle.
Status can be unconditionally reset by instruction ACACC 0.
Status can be unconditionally set by instruction ACYY 0.

2.20 P-Multiplexer

The P-Multiplexer part of the ALU selects data from the Y-REG, the CKB, the RAM, or the DAM, and feeds this data to the Adder-Comparator part of the ALU. (See Figure 1-5)

2.21 N-Multiplexer

The N-Multiplexer part of the ALU selects data from the CKB, the Accumulator, or the RAM, and feeds this data to the Adder-Comparator portion of the ALU. (See Figure 1-5)

2.22 Adder-Comparator

The Adder-Comparator portion of the ALU simultaneously adds and compares the data in the N and P Multiplexers. Information from the comparator is used in determining status. Data from the Adder can be sent to the Accumulator or the Y-REG, or can be dumped if the data is needed for comparison only.

2.23 Accumulator and Y-Register

The Accumulator is a four-bit register that gets its content from the Adder-Comparator. The Y-Register (Y-REG) holds the RAM word address and is used to select one of sixteen Digit latches.

3 Analog/Digital-Converter (ADC)

The A/D-Converter is specially designed for:
- 1-2 KOhm-temperature Si-sensors
- Pt100/Pt500/Pt1000 Elements in conjunction with internal current source
- Silicon pressure sensors

It is in no way restricted to those applications.

The A/D-Converter compares the external analogue input voltage with an internal voltage.

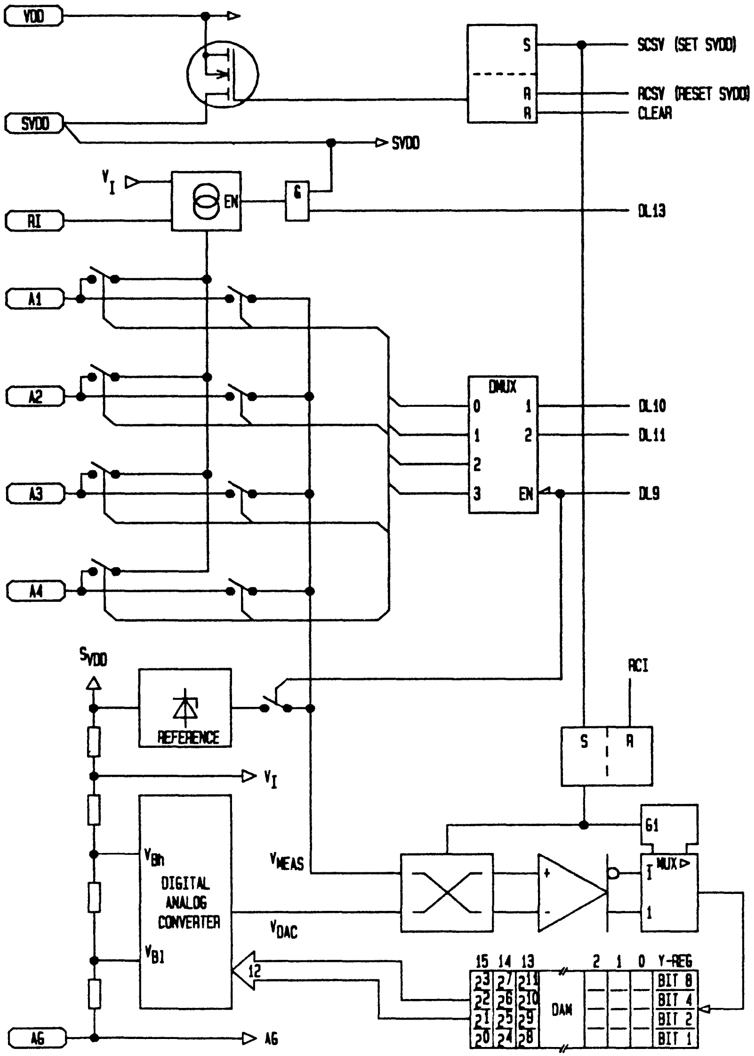


Figure 3-1: Principle of A/D-Converter

3.1 Description of Conversion

The A/D-Conversion is done by executing a software program.

Five instructions are added to the instruction set for controlling the ADC:

- SCSV: switch converter's supply voltage on
- RCSV: reset converter's supply voltage
- RCI: reset converter's comparator inputs to inverted
- TCTM n: transfer comparator (output) to memory - selected by X-REG and Y-REG while n identifies the bit - and set the next lower bit to high (if n is 1, 2, 3).
- TRTM n: transfer range to memory bit n (n = 0, 1,2,3).

The A/D-converter consists of:

- a D/A converter (DAC) as voltage reference source
- a Comparator
- an analog switch to interchange input signals at comparator
- an analog multiplexer (4 to 1) to select one input (A1 - A4)
- a constant current source; programmable by one external resistor and enabled or disabled by DL13
- the SV_{DD} control
- a voltage source (for battery check)

As shown in the Figure 3-1, the instruction SCSV switches the supply voltage to the A/D Converter and PIN SV_{DD} on - instruction RCSV switches it off. The voltage is also switched-off during Power on/$\overline{\text{INIT}}$ and if the processor is in DONE/OFF mode.

The D/A converter is controlled by 12 bits. These 12 bits are memory bits located in the DAM.

DAM

Y-REG	BIT3	BIT2	BIT1	BIT0
13	2^{11}	2^{10}	2^9	2^8
14	2^7	2^6	2^5	2^4
15	2^3	2^2	2^1	2^0

The content of the 12 DAM bits is linearly converted into an analogue voltage by the DAC. This output voltage is fed into the analogue cross bar switch in front of the comparator. The comparator compares this voltage with the analogue input voltage at the selected input A1 ... A4.

A special instruction TCTM is available to do a simple successive approximation by software. This instruction transfers the comparator output to the pre-selected DAM-bit and sets the next lower bit at the same DAM address.

The starting value should be >800 (midpoint of range). The time constant of the DAC network requires some wait cycles between the TCTM instructions:

affected bit	instructions before TCTM
MSB	5
MSB-1	5
MSB-2	5
MSB-3	5
MSB-4	4
MSB-5	4
MSB-6	4
MSB-7	4
MSB-8	3
MSB-9	3
MSB-10	3
LSB	3

Instruction SCSV sets the analog crossbar switch at the comparator input and instruction RCI resets it. Two conversions - one with set and one with reset switch - eliminate the offset of the comparator. The two conversion results are added. The sum is twice the value of the conversion formula.

To ensure that the measurement is within the valid range, a special instruction TRTM is included. This instruction sets the addressed RAM bit to "0" if the measured value is greater than >000 and less than >FFF. Otherwise the bit is set ("1"): selected analog input voltage is out of A/D-conversion range.

Optionally, there are four conversion ranges for the ADC available. One and only one must be selected. This selection is done in connection with the ROM programming.

Conversion formula: (for nominal values)

with: all voltages are referenced to AG

$$\boxed{Vin = (a + k*N)* SVDD} \Rightarrow \boxed{N = \frac{1}{k} * \frac{Vin}{SVDD} - \frac{a}{k}}$$

$$\boxed{\text{with: } 1_{16} \leq N \leq FFE_{16}}$$

single measurement

$$\boxed{\text{with: } N_{LL} \leq N \leq N_{HL}}$$

measurement with interchanged inputs at Comparator inputs (using SCSV / RCI)

Range large	: a = 0.101213203;	k = 0.000096090233 *N
Range small	: a = 0.231271438;	k = 0.000043048228 *N
Range medium-low	: a = 0.1002948411;	k = 0.000073816955 *N
Range medium-high	: a = 0.2326179386;	k = 0.000065411591 *N

Note: N_{LL} = digital value analog input voltage lower limit.
N_{HL} = digital value analog input voltage higher limit.

3.2 Battery Check

The stable internal voltage source (V_{REF}) is applied to the comparator instead of an external voltage signal at the analog inputs. To the other input of the comparator, the DAC output voltage is applied. This DAC output (V_{DAC}) is a linear function of SV_{DD}, that is nearly identical to V_{DD}:

$$V_{DAC} = (a + K \cdot N) \cdot SV_{DD}$$

The comparator allows to determine the N value for which $V_{DAC} = V_{REF}$ and it follows:

$$SV_{DD} = V_{DD} = \frac{V_{REF}}{a \cdot K \cdot N}$$

3.3 Current Source

The current source and the A/D converter are ratiometric. Voltage used for A/D conversion and the reference voltage (V_I) used to set the current of the current source are proportional to SV_{DD} and have a fixed ratio to each other. This ensures optimum tracking. The current source is activated with DL13 set AND with SV_{DD} on; both conditions are necessary to switch on the current source.

The current, passed via an analog input to a sensor, is calculated:

$$I_{An} = \frac{V_{Rext}}{R_{ext}}$$

The current, flowing through a sensor (e.g. resistive temperature sensor), generates a voltage drop at the sensor which is available at the analog input for measurement with the A/D converter:

$$V_{in} = I_{An} \cdot R_{in} \quad \text{with } R_{in} \text{ is sensor's resistance}$$

$$V_{in} = V_{Rext} \cdot \frac{R_{in}}{R_{ext}}$$

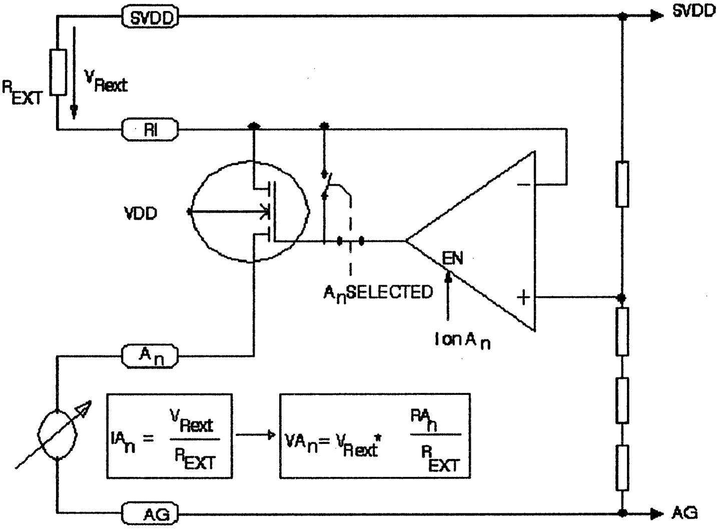

SVDD
SVDD
R_{EXT}
V_{Rext}
RI
VDD
A_nSELECTED
EN
I on A_n
A_n
$IA_n = \frac{V_{Rext}}{R_{EXT}}$
$VA_n = V_{Rext} * \frac{RA_n}{R_{EXT}}$
AG
AG

Figure 3-2: Principle of Current Source

DL13	DL11	DL10	DL9	analog input selected, Current Source (CS) on/off
0	0	0	0	A1, CS off
0	0	1	0	A2, CS off
0	1	0	0	A3, CS off
0	1	1	0	A4, CS off
1	0	0	0	A1, CS on
1	0	1	0	A2, CS on
1	1	0	0	A3, CS on
1	1	1	0	A4, CS on
X	X	X	1	battery check, CS off

Figure 3-3: Current Source

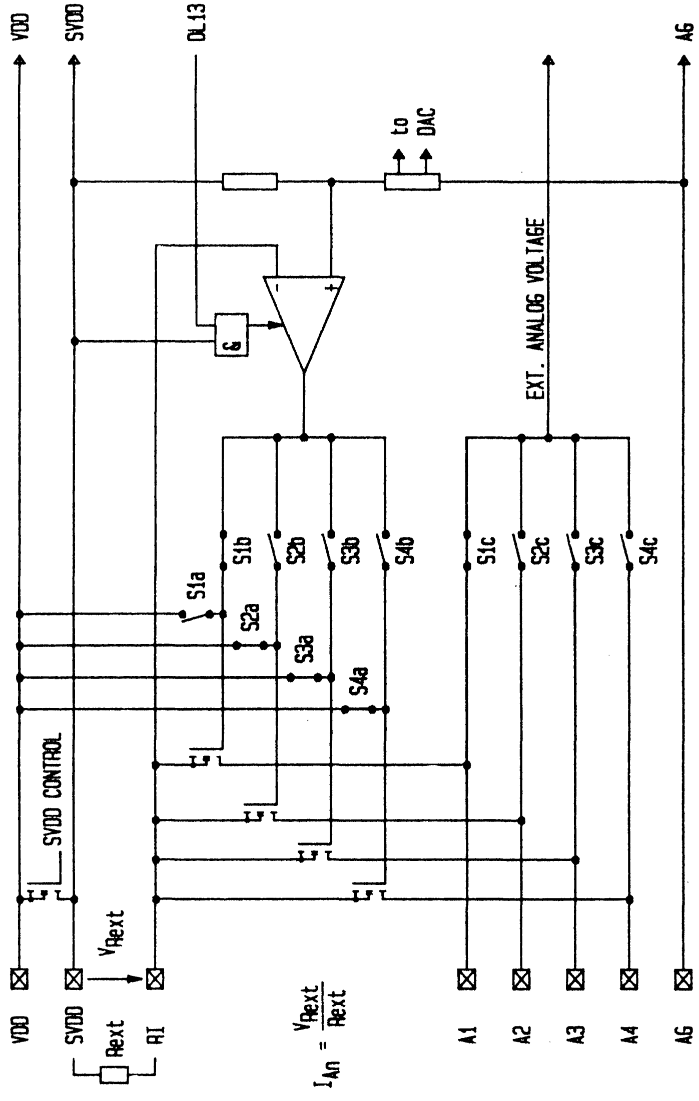


Figure 3-3: Current Source (continued)

4 Display

4.1 Gate Level PLA

The TSS400/4 has a 1 of 64 decode for character output. The decode has six input lines; from the Most Significant Bit (MSB) to the Least Significant Bit (LSB) they are:

- DL14
- Status bit
- The four Accumulator bits from ACC8 to ACC1.

The decoded information is then transferred to a bus to the Segment PLA. The Segment PLA can be gate-level programmed to set the state of each of the seven segment lines, A-G, high or low depending on the decode information.

The Segment H is controlled by SDP/RDP instruction.

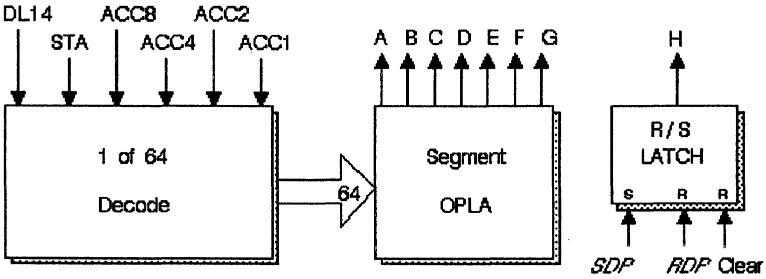


Figure 4-1: Decode and OPLA for Segment Output

4.2 Common Select Lines

There are four Common Select Lines. Each has a unique output waveform that when used in combination with the twenty (20) Select Lines can control 40/80 different segments on a liquid crystal display (LCD).

The combination of Common Select Lines to Segment Lines is shown in the next paragraph.

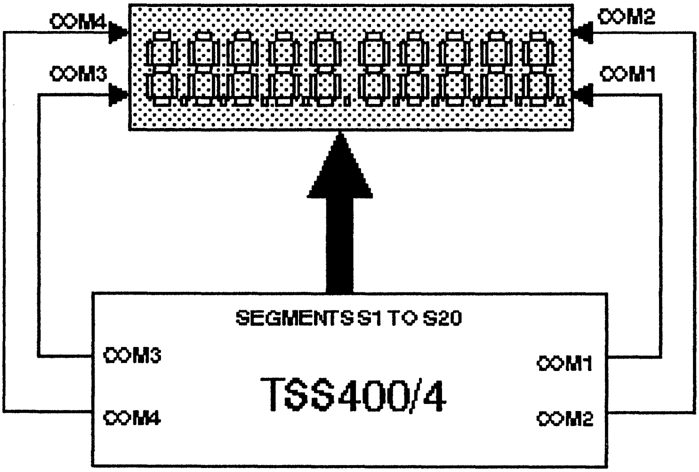


Figure 4-2: 4Mux LCD Application Example

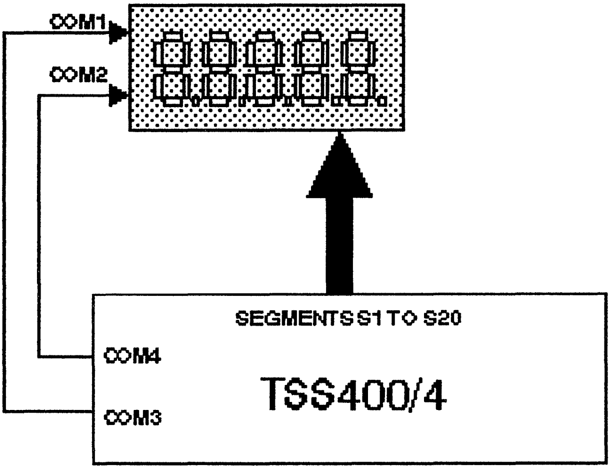


Figure 4-3: 2Mux LCD Application Example

4.3 Select Lines

Up to twenty Select Lines are available as outputs. Each Select Line defines two/four Segments as shown below. A 64-term gate programmable-segment PLA (decoded by ACC1, ACC2, ACC4, ACC8 Status, DL14) and SGH determines which segments are to be turned on.

These are then loaded into the Segment Latches by Y-Decoder/TDO instruction and time-multiplexed to the Output Select Lines.

TSS400/4 Segment Latch Structure

(Standard Common-Segment-Y option)*).

1/2 duty cycle (2 mux)

Select Line	Segments	Loaded by
	C C	
	O O	
	M M	
	3 4	
S1	B H	Y0/TDO
S2	A F	Y0/TDO
S3	G E	Y0/TDO
S4	C D	Y0/TDO
S5	B H	Y1/TDO
S6	A F	Y1/TDO
S7	G E	Y1/TDO
S8	C D	Y1/TDO
S9	B H	Y2/TDO
S10	A F	Y2/TDO
S11	G E	Y2/TDO
S12	C D	Y2/TDO
S13	B H	Y3/TDO
S14	A F	Y3/TDO
S15	G E	Y3/TDO
S16	C D	Y3/TDO
S17	B H	Y4/TDO
S18	A F	Y4/TDO
S19	G E	Y4/TDO
S20	C D	Y4/TDO

1/4 duty cycle (4 mux)

Select Line	Segments	Loaded by
	C C C C	
	O O O O	
	M M M M	
	1 2 3 4	
S1	A B C D	Y0/TDO
S2	F G E H	Y0/TDO
S3	A B C D	Y1/TDO
S4	F G E H	Y1/TDO
S5	A B C D	Y2/TDO
S6	F G E H	Y2/TDO
S7	A B C D	Y3/TDO
S8	F G E H	Y3/TDO
S9	A B C D	Y4/TDO
S10	F G E H	Y4/TDO
S11	A B C D	Y5/TDO
S12	F G E H	Y5/TDO
S13	A B C D	Y6/TDO
S14	F G E H	Y6/TDO
S15	A B C D	Y7/TDO
S16	F G E H	Y7/TDO
S17	A B C D	Y8/TDO
S18	F G E H	Y8/TDO
S19	A B C D	Y9/TDO
S20	F G E H	Y9/TDO

*) Note: The Y-Address of each Segment Group is configurable to application needs. The definition is done together with the ROM programming (via mask).

4.4 Display Drive Waveforms

Display-Drive waveforms are shown in Figure 4-4 for 1/2 and 1/4 duty cycle LCD. The internal logic signal KSC initiates voltage changes in the display voltages.

KSC has a frequency of 512 Hz and is active for 122 micro-seconds (f_{TOSCIN} = 32768 Hz).

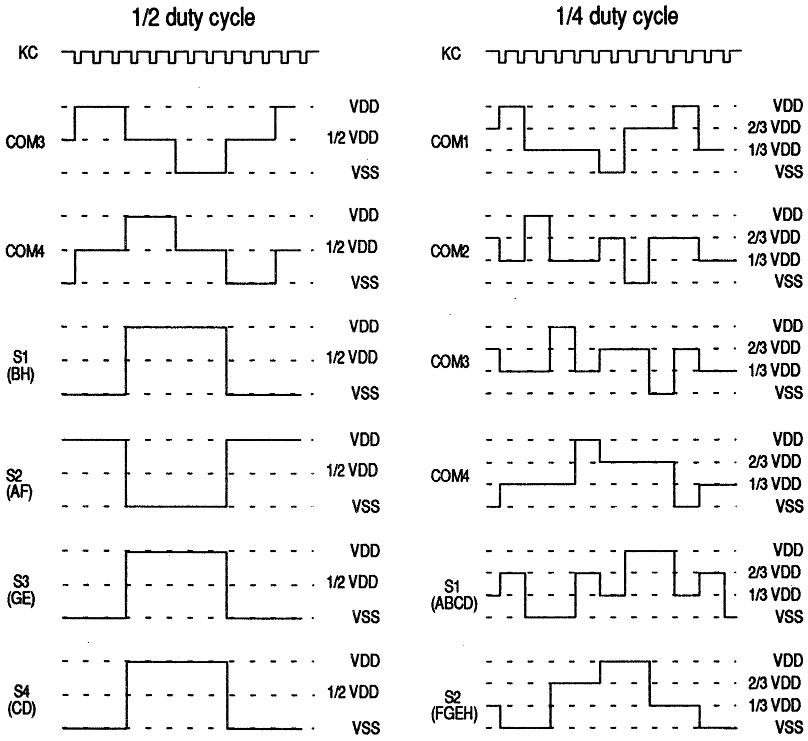


Figure 4-4: Display Drive Waveforms (Standard Gate Placement) e.g. Number 4: Segment 'BCFG' on - 'ADEH' off

5 Instruction Set Description

The following abbreviations will be used in describing the Instruction Set:

ACC	= Accumulator
CONSTANT	= 0-15
LSB	= Least Significant Bit
MEM	= Memory as pointed to by X and Y Registers
MSB	= Most Significant Bit
REG	= Register
SS	= Special Status Latch
<>	= not equal
$\overline{\text{ACC}}$	= not Accumulator

5.1 Functional Description (alphabetical listing)

CODE	MNEMONIC	DESCRIPTION
070-07F	**ACACC**	Add a constant to ACC and store in ACC; if carry status = 1
0F0-0FF	**ACYY**	Add a constant to Y-REG
0E0-EF	**ALEC**	If ACC less than or equal CONSTANT, status = 1
001	**ALEM**	If ACC is less than or equal to MEM, status = 1
015	**AMAAC**	Put MEM + ACC into ACC; if carry, status = 1
200-2FF	**BRANCH**	If status = 1, branch to location >00 to >FF as pointed to by label; else increment Program Counter
300-3FF	**CALL**	If status = 1, then execute CALL to location >00 to >FF as defined by label. The 3-level subroutine stack retains the return address.
012	**CCLA**	If SS = 0, put 0 in ACC; if SS = 1, put 1 in ACC
006	**CLA**	Clear ACC
0B2	**COMX8**	Complement MSB of X-REG
031	**CPAIZ**	Put 2's complement of ACC into ACC; if initial ACC = 0, status = 1
018	**CTMDYN**	If both Add Latch and SS are set or both are reset, put ACC into MEM; Decrement Y-REG; if initial Y <> 0, status = 1
007	**DMAN**	Put MEM-1 into ACC; if MEM <> 0, status = 1

5.1 **Functional Description** (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
010	**DMEA**	Put DAM + MEM + SS into ACC
011	**DNAA**	Add DAM, $\overline{\text{ACC}}$ and SS and store it in ACC
0BE	**DONE**	Turns the processor portion of the chip off
004	**DYN**	Decrement Y-REG; if initial Y <> 0 then status = 1
032	**IMAC**	Store in ACC the result of MEM +1; if MEM = 15 then status = 1
01A	**IOIN**	I/O Output buffer becomes 3-state to be Input for ext. signals
005	**IYC**	Increment Y-REG; if initial Y = 15 then status = 1
00E	**KNEZ**	If K-lines <> 0 then status = 1. If status is set to 0 then another KNEZ 6-instruction cycles later must be done; if status = 0 again you can be certain that K-lines = 0.
080-08F	**LDP**	Load Constant into ROM page buffer
090-09F	**LDX**	Load Constant into X-REG. I(5) = LSB, I(8) = MSB. RESET DAM-LATCH
009	**MNEA**	If MEM <> ACC then status = 1
033	**MNEZ**	If MEM <> 0 then status = 1

5.1 **Functional Description** (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
013	**NDMEA**	Add MEM, $\overline{DAM}$ and SS and store it in ACC
0B7	**OFF**	Turns the Processor and Timer off
014	**ORMA**	Logical OR of MEM and $\overline{ACC}$ stored in ACC
017	**ORMNAA**	Logical OR of MEM and $\overline{ACC}$ stored in ACC
0A4-0A7	**RBIT**	Reset one of 4 MEM bits as addressed by I(8), I(9) LSB- Bit 2^0 → @I(7), @I(8) Bit 2^1 → I(7), @I(8) Bit 2^2 → @I(7), I(8) MSB- Bit 2^3 → I(7), I(8)
0BD	**RCI**	Reset Comparator offset switch to INVERTED
0BB	**RCSV**	Reset Converters Supply Voltage and Pin SV$_{DD}$
0B6	**RDP**	Reset Decimal Point (Segment H)
0B4	**REAC**	Reset END-Around-Carry (SS)
0BF	**RETN**	Reset add latch and branch latch. When in CALL, RETN also transfers subroutine register to Program Counter and transfers ROM page register to Page Buffer + Page Address

5.1 **Functional Description** (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
00B	**RSTIO**	Reset I/O pin - I/O Output buffer is enabled
036	**RSTR**	Resets Digit Latch pointed to by Y-REG
0B1	**SAL**	Set ADD latch
030	**SAMAN**	MEM + $\overline{ACC}$ +1 → ACC; if MEM ≥ ACC then status = 1
0B3	**SBL**	Set Branch Latch
0A0-A3	**SBIT**	Set 1 of 4 MEM bits as addressed by I8 and I9
0BA	**SCSV**	Set Converters Supply Voltage and Pin SV_{DD}. The Comparator offset is switched to NONINVERTED
0B9	**SDP**	Set Decimal Point (Segment H)
0B5	**SEAC**	Set End-Around-Carry (SS)
00C	**SETIO**	Set I/O - I/O Buffer is enabled
00D	**SETR**	Set Digit Latch pointed to by Y-REG
0B8	**TAK**	Transfer ACC to K-latch for Output
03F	**TAM**	Transfer ACC to MEM
0D0-DF	**TAMACS**	Transfer ACC to MEM; Add a constant to ACC and store in ACC; if Add Latch is set and carry from ALU on previous or present instruction is = 1, set SS; reset otherwise; if Add Latch is reset and if carry from ALU on only previous instruction = 1, set SS
03C	**TAMDYN**	Transfer ACC to MEM; decrement Y-REG; if initial Y <> 0, status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
03D	**TAMIYC**	Transfer ACC to MEM; Increment Y-REG, if initial Y = 15, status = 1
03E	**TAMZA**	Transfer ACC to MEM; clear ACC
038	**TAY**	Transfer ACC to Y-REG
020-023	**TBIT**	Test 1 of 4 MEM bits as addressed by I8 and I9. If MEM bit EQ 1, status = 1
0A8-0AB	**TBITA**	Test 1 of 4 ACC bits as addressed by I8 and I9. If ACC bit EQ 1, status = 1
0C0-0CF	**TCA**	Transfer a Constant to ACC
040-04F	**TCY**	Transfer a Constant to Y-REG
060-06F	**TCMIY**	Transfer a Constant to MEM; increment Y-REG
024-027	**TCTM**	Transfer Comparator to 1 of 4 MEM bits as addressed by I8 and I9, the next lower nibble at the same is set.
016	**TDA**	Transfer DAM to ACC independent of X
0BC	**TDL**	Toggle DAM Latch
0B0	**TDO**	Load the Segment Latches with the segment data (A-G) and decimal point into the latches being addressed by Y (0-4/9)
0AC-0AF	**TIOM**	Transfer I/O to 1 of 4 MEM bits as addressed by I8 and I9 (Level sensed during previous instr.)
008	**TKA**	Transfer K-lines to ACC
00A	**TKM**	Transfer K-lines to MEM
039	**TMA**	Transfer MEM to ACC
01D	**TNEA**	If Timer <> ACC, status = 1
01C	**TNEM**	If Timer <> MEM, status = 1

5.1 Functional Description (alphabetical listing - continued)

CODE	MNEMONIC	DESCRIPTION
03A	**TMY**	Transfer MEM to Y-REG
028-02B	**TRTM**	Transfer Range (ADC) to 1 of 4 MEM as addressed by I8 and I9 (ADC in Range: 0; out of Range: 1)
00F	**TSIO**	Transfer Status to I/O - I/O Buffer is enabled
01E	**TTA**	Transfer seconds to ACC if DL12 = 0; transfer 1/16 seconds to ACC if DL12 = 1
01B	**TTM**	Transfer seconds to MEM if DL12 = 0; transfer 1/16 seconds to MEM if DL12 = 1
01F	**TTY**	Transfer seconds to Y-REG if DL12 = 0; transfer 1/16 seconds to Y-REG if DL12 = 1
03B	**TYA**	Transfer Y-REG to ACC
019	**XDA**	Exchange DAM and ACC
003	**XMA**	Exchange MEM and ACC
002	**YNEA**	If Y-REG <> ACC, status = 1
050-05F	**YNEC**	If Y-REG <> Constant, status = 1

5.2 10-Bit Instruction Map

<table>
<tr><th></th><th colspan="8">LSB</th></tr>
<tr><th>MSB</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>00</td><td></td><td>ALEM</td><td>YNEA</td><td>XMA</td><td>DYN</td><td>IYC</td><td>CLA</td><td>DMAN</td></tr>
<tr><td>01</td><td>DMEA</td><td>DNAA</td><td>CCLA</td><td>NDMEA</td><td>ORMA</td><td>AMAAC</td><td>TDA</td><td>ORMNAA</td></tr>
<tr><td>02</td><td colspan="4">TBIT</td><td colspan="4">TCTM</td></tr>
<tr><td>03</td><td>SAMAN</td><td>CPAIZ</td><td>IMAC</td><td>MNEZ</td><td></td><td></td><td>RSTR</td><td></td></tr>
<tr><td>04</td><td colspan="8">TCY</td></tr>
<tr><td>05</td><td colspan="8">YNEC</td></tr>
<tr><td>06</td><td colspan="8">TCMIY</td></tr>
<tr><td>07</td><td colspan="8">ACACC</td></tr>
<tr><td>08</td><td colspan="8">LDP</td></tr>
<tr><td>09</td><td colspan="8">LDX</td></tr>
<tr><td>0A</td><td colspan="4">SBIT</td><td colspan="4">RBIT</td></tr>
<tr><td>0B</td><td>TDO</td><td>SAL</td><td>COMX8</td><td>SBL</td><td>REAC</td><td>SEAC</td><td>RDP</td><td>OFF</td></tr>
<tr><td>0C</td><td colspan="8">TCA</td></tr>
<tr><td>0D</td><td colspan="8">TAMACS</td></tr>
<tr><td>0E</td><td colspan="8">ALEC</td></tr>
<tr><td>0F</td><td colspan="8">ACYY</td></tr>
<tr><td>10 to 1F</td><td colspan="8">DISABLED</td></tr>
<tr><td>20 to 2F</td><td colspan="8">BRANCH</td></tr>
<tr><td>30 to 3F</td><td colspan="8">CALL</td></tr>
</table>

Note: e.g.	SETR	: fixed instruction	(28)
	+ XDA	: micro/fixed instruction	
	other	: micro instructions	(48)

5.2 10-Bit Instruction Map (Continued)

	LSB							
MSB	8	9	A	B	C	D	E	F
00	TKA	MNEA	TKM	RSTIO	SETIO	SETR	KNEZ	TSIO
01	CTMDYN	+ XDA	IOIN	TTM	TNEM	TNEA	TTA	TTY
02	TRM							
03	TAY	TMA	TMY	TYA	TAMDYN	TAMIYC	TAMZA	TAM
04	TCY							
05	YNEC							
06	TCMIY							
07	ACACC							
08	LDP							
09	LDX							
0A	TBITA				TIOM			
0B	TAK	SDP	SCSV	RCVS	TDL	RCI	DONE	RETN
0C	TCA							
0D	TAMACS							
0E	ALEC							
0F	ACYY							
10 to 1F	DISABLED							
20 to 2F	BRANCH							
30 to 3F	CALL							

Note: e.g. SETR : fixed instruction (28)
+ XDA : micro/fixed instruction
other : micro instructions (48)

6 Electrical Description

6.1 Absolute Maximum Ratings

Voltage applied at V_{DD} to V_{SS}	- 0.3 V to + 7.0 V
Voltage applied to any pin (referenced to V_{SS})	V_{SS} -0.3 V to V_{DD}+0.3 V
Diode current at any device terminal	+/- 2 mA
Storage Temperature	- 55 °C to 150 °C

6.2 TSS400/4 - Operating Conditions (short form)

Parameter:	MIN	NOM	MAX	UNITS
Supply Voltage, V_{DD}	2.6	3.0	5.5	V
Supply Voltage, V_{SS}	0.0	0.0	0.0	V
Operating Temperature				
Range I	-25	27	+85	°C
Range A	-40	27	+125	°C
Timer Frequency (XTAL)		32.768		KHz
*)Timer Frequency (ext)	20		100	KHz
Processor Frequency internal (f_{proc})		700		
Characteristics: see Figure 6-1				
Processor Frequency external,				
PFRQ Pin (f_{proc})	400	700	1000	KHz
**)Instruction execution time	15	8.5	6	µSec
Capacitance of Display:				
any Common			1000	pF
any Segment			200	pF

*) Note: If LCD-display is used be sure that the frequency meets

the limits of display frame rate $\left(DFR = \frac{\text{timer frequency}}{512}\right)$

**) Note: Each instruction needs 6 cycles of the

processor frequency $f_{proc} = t_{instruct} = \frac{6}{f_{proc}}$

The technical parameter are the same as mentioned in the TSS400 User's Guide. Please refere to this. The same derivates as with the TSS400 are available with the TSS400/4.

6.3 Typical RC-MOS Oscillator Characteristics

Not available yet.

Figure 6-1:

6.4 Schematics of Inputs/Outputs

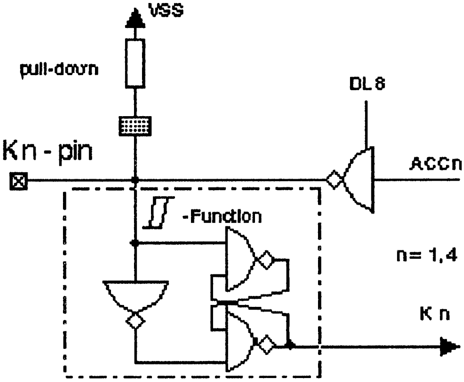


Figure 6-2: Schematic of K-I/Os

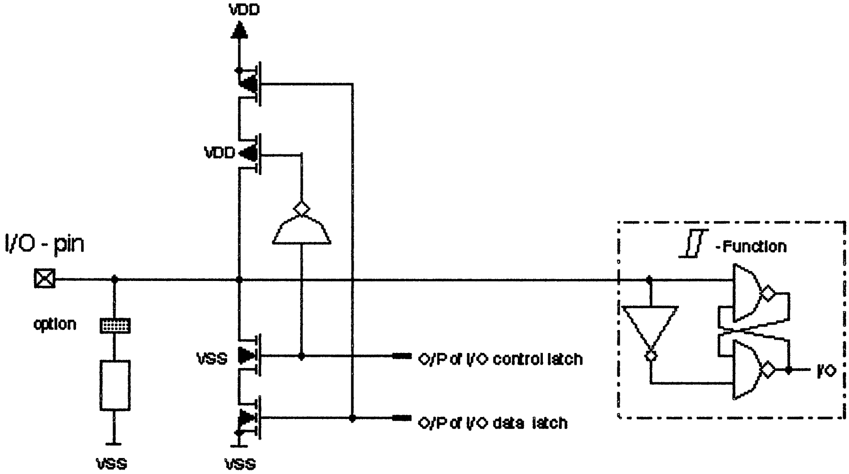


Figure 6-3: Schematic of I/O

6.4 Schematics of Inputs/Outputs (continued)

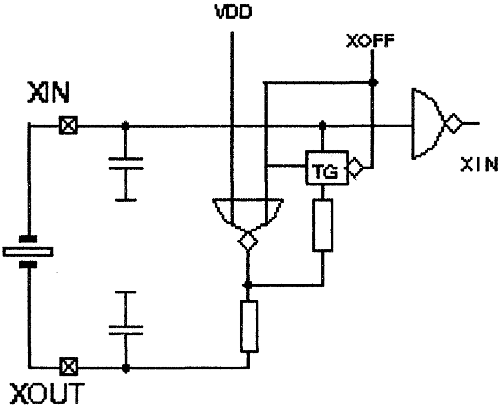


Figure 6-4: Schematic of XTAL Oscillator

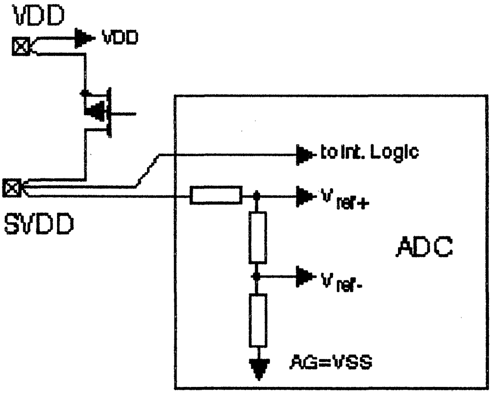


Figure 6-5: Schematic of SVDD

APPENDIX A - Software Restrictions

1. Carefully examine the states of inputs and outputs when DONE or OFF instruction is used. Prevent additional DC current in low power consumption mode caused by the states of inputs and outputs.

2. Pay attention to the states of I/O pin and the K-lines while being in DONE/OFF. The wake-up can be triggered by K1, K2, K4, K8 and I/O - but only if all were Low before one of these Lines goes High.

3. The processor begins execution at page 15; PC=00 the first time power is applied. Each time a "wake-up" signal occurs, the processor begins execution at page 15; PC = >FF.

4. For operations which require more then 15 seconds of processing, the software should periodically update the timekeeping since the chip provides only modulo-16-sec.

5. The SAL instruction may cause the End-Around-Carry (SS) to change state falsely in either direction. Thus, after an SAL, the SS should be set to the desired state with a SEAC or REAC. Also, if both the Add latch and SS must be set to a desired state, the SAL should be done first. For example, in the basic Add loop where both the Add latch must be set and the SS reset, the SAL should be executed prior to the REAC.

6. The TTA and TTM instructions:
   It is possible to get a false intermediate reading because of the Timer and Processor operate asynchronous. To prevent this, the time check must be repeated until two successive tests yield the same value.

7. Between a wake-up from the timer and a DONE instruction there must be a minimum number of instructions N:

$$N > = \frac{4 \text{ x processor frequency}}{6 \text{ x } f_{TOSCIN}} + 1$$

8. To execute a three instruction BCD addition, the following sequence of instructions may be used:

LOC		INSTRUCTION		COMMENT
001		SBL		Before jumping to subroutine, set Branch Latch
002		CALL	START	
"		"		
"		"		
"		"		
010	ADDBCD	TAMACS	6	
011		CTMDYN		
012	START	BRANCH	ADDBCD	Branch to location 10 and perform a DMEA instruction (code 10) before executing TAMACS
013		CCLA		
014		RETN		

9. SBL instruction:
If Branch Latch is set use only Branches which can operate on an instruction. In other ways it is possible that some of the micro-instructions are active and will cause some problems (i.e., transfer CKB-Bus to memory...). Don't forget to reset Branch Latch after using it.

10. Processor enable logic "wake-up"
Between TKA and DONE/OFF instruction not more than three instructions are allowed, if a successful "wake-up" should be performed by external hardware (→ K inputs / I/O).

11. Processor logic "wake-up" enable
    If the option of "wake-up" the processor by the negative and/or positive edge of K8 is used, pay attention to the effect: inverting K8 by DL14 can produce an edge. If this happens less than 7 instructions before the DONE instruction the processor could "wake up" due to the integrated delay line.

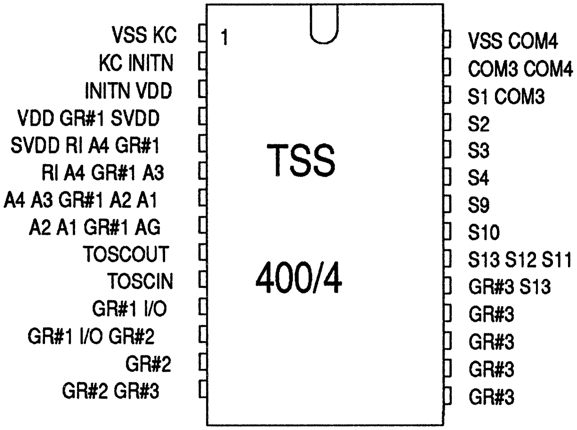
12. Status after "wake-up"

    Register content after entry into active Processor-Mode

Register		**Power up**	**INITN pin**	**DONE Mode**	**OFF Mode**
Program Counter	PC	00	00	FF	FF
Page Address	PA	F	F	F	F
Page Buffer	PB	F	F	F	F
Status	S	undefined	undefined	undefined	undefined
Special Status	SS	undefined	undefined	undefined	undefined
Accumulator	A	undefined	undefined	undefined	undefined
YRegister	Y	undefined	undefined	undefined	undefined
XRegister	X	undefined	undefined	undefined	undefined
RAM Content		undefined	unchanged	unchanged	unchanged
DAM Latch		undefined	undefined	undefined	undefined
ADD Latch		reset	reset	reset	reset
BRANCH Latch		reset	reset	reset	reset
SEG-H Decimal point	DP	reset	reset	reset	reset
Digit Latches	DLn	reset	reset	unchanged	unchanged
K-Port Data Latch		undefined	unchanged	unchanged	unchanged
I/O Port Data Latch		undefined	unchanged	unchanged	unchanged
I/O Port Control Latch		reset	reset	unchanged	unchanged
Timer0 and Timer1		undefined	unchanged	actual Time	undefined
ADC/Pin Supply SVDD		switched off	switched off	switched off	switched off
LCD Segment Latches		undefined	unchanged	unchanged	unchanged

APPENDIX B - Pinning Selection Schematic

28 PIN DUAL-IN-LINE PLASTIC PACKAGE



Note:

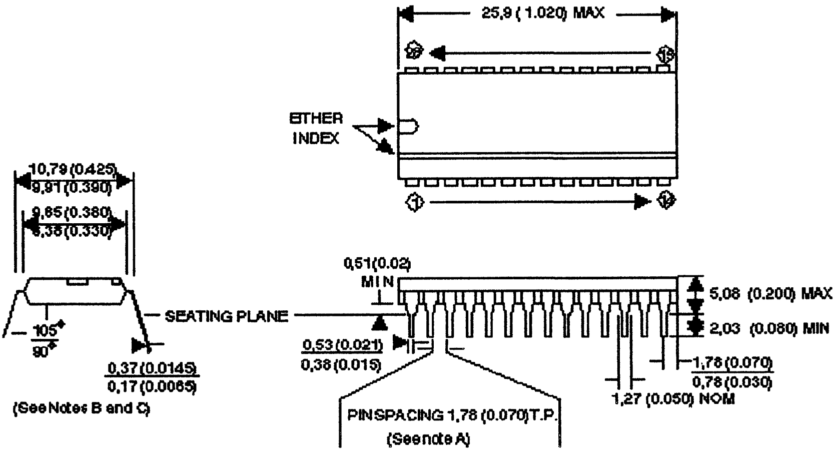
mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

APPENDIX B - Pinning Selection Schematic (continued)

28-pin NF package (YS)



All dimensions are in millimeter and parenthetically in inches.

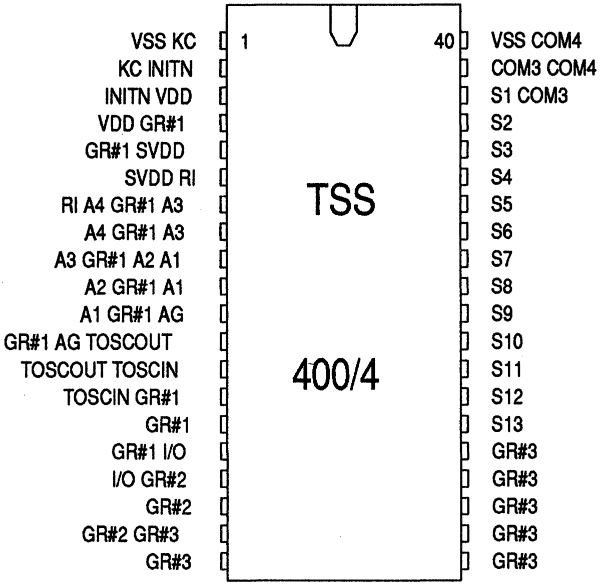
Notes: A: Each pin centerline is located within 0,25 (0.010) of its true longitudinal position.

B: This dimension does not apply for solder-dipped leads.

C: When solder-dipped leads are specified, dipped area of the lead extends from the lead tip to at least 0,51 (0.020) above seating plane.

APPENDIX B - Pinning Selection Schematic (continued)

40 PIN DUAL-IN-LINE PLASTIC PACKAGE



Note:

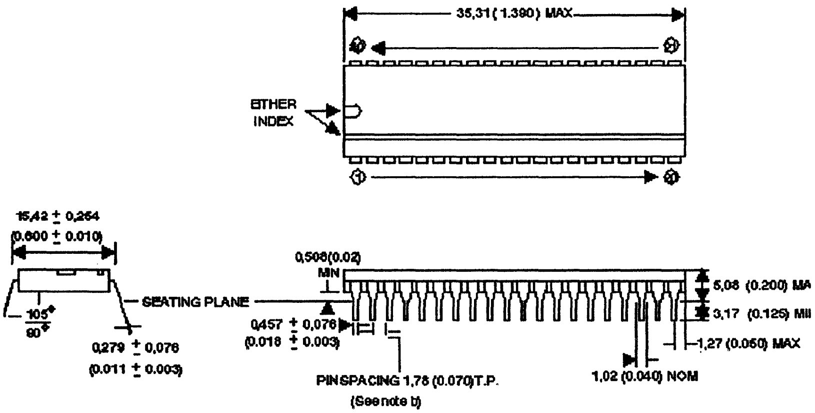
mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

APPENDIX B - Pinning Selection Schematic (continued)

40-pin N2 package (YS)

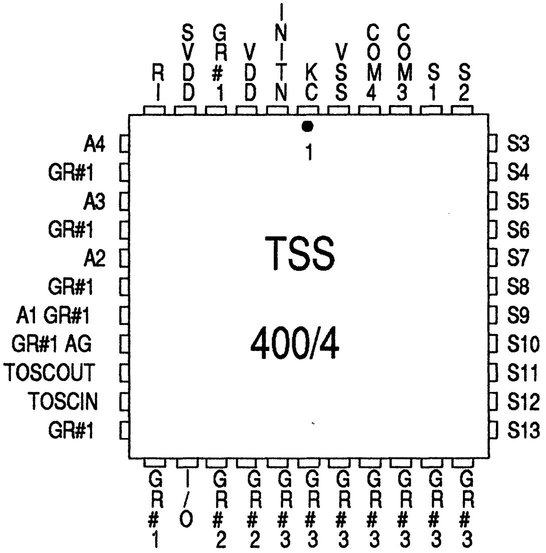


Notes: A: All dimensions are in millimeter and parenthetically in inches.

B: Each pin centerline is located within 0,26 (0.010) of it's true longitudinal position.

APPENDIX B - Pinning Selection Schematic (continued)

44 PIN PLASTIC LEADED CHIP CARRIER PLCC



Note:

mandatory signals are:
VDD,VSS,KC,INITN,SVDD,
TOSCIN,TOSCOUT,S2,S3,S4,S9,S10,S11

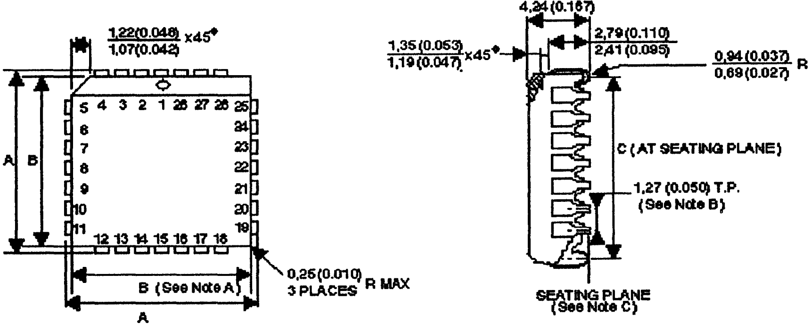
GR#1 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
GR#2 = R0 R1 R2 R3 K1 K2 K4 K8 CLOCK PFRQ
COM2 COM1 S20 S19 S18 S17 S16 S15 S14
GR#3 = COM2 COM1 S20 S19 S18 S17 S16 S15 S14
R4 R5 R6 R7

Note: NC - No internal connection NC pins may be used for test purposes in agreement with customer.

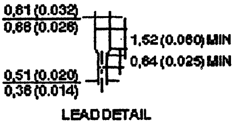
APPENDIX B - Pinning Selection Schematic (continued)

FN Plastic Chip Carrier Package

(28-terminal package used for illustration)



JEDEC Outline	No. of terminals	A min.	A max.	B min.	B max.	C min.	C max.
MO-047AC	44	17,40 (0.685)	17,65 (0.695)	16,51 (0.650)	16,66 (0.656)	15,49 (0.610)	16,00 (0.630)



Notes: A: Centerline of center pin each side is within 0,10 (0.004) of package centerline as determine by dimension B.

B: Location of each pin is within 0,127 (0.006) of true position with respect to center pin on each side.

C: The lead contact points are planar within 0,10 (0.004).

All dimensions and notes for the specified JEDEC outline apply.

All linear dimensions are in millimeters and parenthetically in inches.

APPENDIX C - User Options

LCD

- Segment PLA with 64 terms:
  4bit ACCU + Status + DL14 define on/off of segments A to G. Segment H is defined by instructions SDP and RDP.
- Common lines
  2MUX (1/2 duty) or 4MUX (1/4 duty). (Some manufacturer use back plane instead of common).
- Segment lines
  1. Attachment segment to common.
  2. Attachment segment to Y-Register.

Note: All three options must be defined even if no display will be used.

WAKE-UP FREQUENCY

- DL12 = 0 : 1 Hz (no option)
- DL12 = 1 : 128 OR 64 OR 32 OR 16 OR 8 Hz only one must be selected.

PROCESSOR FREQUENCY

- Internal RC-MOS Oscillator (see Figure 6-1).
- External Supply on Pin PFRQ (see 6.3 Operating Conditions)

Note: By using the "EXTERN" Option the internal Osc can be overwritten on Pin "PFRQ".
External Signal strength required >10 µA @ V_{DD}-0,4 V, V_{SS}+0,4 V

ADC RANGE

- SMALL
- MEDIUM LOW
- MEDIUM HIGH
- LARGE

Note: One and only one must be selected.

TIMOSC

- TRC RC Oscillator
- TXOSC Christal Oscillator

If the RC Oscillator is selected, a resistor needs to be connected from TOSCIN to TOSCOUT and a capacitor from TOSCIN to VSS. Please make sure that the frame frequency is o.k. if an LCD is used.

PULL-DOWN

Individually selectable for K1, K2, K4, K8 AND I/O.

Kn OPEN-SOURCE (n = 1,2,4,8)

Individually selectable for K1, K2, K4 and K8 to be open source

CLOCK

Frequency output divided from the crystal oscillator (XCOSIN).

- NONE : nothing selected - no (bond)pad used
- 256 Hz : $f_{CLOCK} = f_{TOSCIN}/128$
- 1024 Hz : $f_{CLOCK} = f_{TOSCIN}/32$

K8 FLIP-FLOP

Signal on K8 line not immediate available for ACCU/MEMORY; if this option is used, signal at K8 pin is divided by 2 and then available for further operations (TKA, TKM, KNEZ). The divider by 2 is reset by PUC (power-up clear) or INIT.

PINNING

Variable pinning helps optimization of system cost including PCB.

TSS400 Family User's Guide

Sensor Signal Processors

Part III

TSS400 Family User's Guide

Sensor Signal Processors

TSS400(/4) Software User's Guide
July 1991

Table of Contents

List of Figures

1 Introduction

1.1 General

This section introduces the TSS400(/4) one-chip sensor signal processor and outlines how an algorithm is developed and implemented to achieve cost effective designs. This introduction includes a definition of terms and conventions. This manual treats the TSS400(/4) as a system of logic and analog blocks controlled by the programmer.

After introduction of the hardware a detailed presentation of the standard instructions will follow. Hints for efficient algorithms and a number of example programs are presented last, in the sections 4 to 6, since they require a thorough understanding of the standard instruction set.

1.2 Design Features

The TSS400(/4) series architecture is constructed to suit a wide variety of applications. The design is both cost effective and flexible because data input, processing and output are performed in one self-contained unit. An internal ROM, RAM A/D converter, ALU and LCD driver comprise a single chip sensor signal processor which functions according to the ROM program and the system inputs.

Features

- minimal system: one device containing ROM, RAM, inputs, outputs, A/D converter, LCD driver, timer and more
- 12 bit A/D converter with 4 selectable inputs and 1 out of 4 ranges.
- programmable current source from 0.15 to 2.4 mA · V_{DD}/V for all A/D converter inputs
- mask programmable 2 or 4 common multiplex LCD driver
  2 common driver controls up to 5 digits
  4 common driver controls up to 10 digits
- 64 character, 7 segment output PLA, decimal point segment is switched independently

- 2048 nine bit words of ROM / 4096 nine bit words of ROM
- 576 bits of static RAM / 960 bits of static RAM
- three levels of subroutine
- timekeeping from an on chip quartz oscillator (32768 Hz) or an RC oscillator
- low power silicon gate CMOS process
- DONE mode for reduction of power consumption with active timer and RAM; the CPU is woken up regularly by 2 selectable frequencies or by input changes
- OFF mode for very low power consumption with active RAM; wakeup by input changes
- internal MOS oscillator or external frequency input selectable
- three instruction BCD addition possible
- easy tailoring to actual needs by several hardware options
- one bit I/O port for fast data exchange
- 3 different packages available

1.3 Design Steps

Several steps are necessary to complete a system consisting of the TSS400(/4) and its periphery with the necessary performance. A typical and recommended development cycle is shown below:

1. Definition of tasks to be performed by TSS400(/4) and periphery.
2. Definition of the needed hardware options (ADC range, LCD hardware, wakeup frequency etc.).
3. Decision if 2 common or 4 common LCD display.
4. Input and output definition (K-Port, R outputs, ADC inputs, segment outputs, I/O pin etc.)
5. Drawing a complete hardware schematic.
6. Worst case design for all external components.
7. RAM organization.

8. Flowcharting the complete program for determining the instruction coding necessary to fulfil specification requirements.
9. Coding of the program with an editor.
10. Assembling the program with TSS400 assembler.
11. Testing the program with the ADT400(/4) emulator after removing of all assembly errors.
12. After approval of the program, preparation of the production definition file with the ADT system (hardware options, ROM code).
13. Sending the specification file to TI for manufacturing prototype circuits.
14. After checking and approving the prototype circuits, TI starts volume production.

1.4 Symbols and Conventions

1.4.1 List of Abbreviations

A	Accumulator Register	
ADC	Analog Digital Converter	
ALU	Arithmetic and Logic Unit	
B	Bit Field of Instruction Word	(2 bits)
C	Constant Field of Instruction Word	(4 bits)
CKB	Constant and K-Lines Bus	
CPU	Central Processing Unit	
DAM	Direct Access Memory	M(15,0-15) or M(7,0-15)
DP	Decimal Point Latch	
DLn	Digit Latch n	(n = 0 - 15)
Kn	K-Port n	(n = 1, 2, 4, 8)
LCD	Liquid Crystal Display	
LSB	Least Significant Bit	
LSD	Least Significant Digit	
MSB	Most Significant Bit	
MSD	Most Significant Digit	
M(X,Y)	RAM Memory Location	
M(X,Y,B)	RAM Memory Bit Location	(B = 0 - 3)

OPLA	Output Programmable Logic Array	
PA	Page Address Register	
PB	Page Buffer Register	
PC	Program Counter	
RAM	Random Access Memory (Read/Write)	
ROM	Read Only Memory	
Rn	R Output n	(n = 0 - 7)
S	Status Bit	
SS	Special Status Bit (End-Around-Carry)	
Timer0	Timer Nibble fed from 1 Hz (DL12 = 0)	
Timer1	Timer Nibble fed from 16 Hz (DL12 = 1)	
W	Branch Address of Instruction Field (7 bits)	
X	RAM X Address Register	
Y	RAM Y Address Register	
SRn	Subroutine PC Register Level n	(n = 1 - 3)
PSRn	Subroutine Page Register Level n	(n = 1 - 3)
10En	10 raised to n-th power	
EMI	Electro Magnetic Interference	

Note: M(X,Y), M(X,Y,B), "Addressed Memory" and "Memory addressed by X and Y register" are used for indicating the currently addressed memory location or memory bit.
This means:
If DAM Latch = 0: X = contents of X register
If DAM Latch = 1: X = 15

1.4.2 Symbols and Logic Notations

a → b	Transfer value a to b
a ↔ b	Exchange contents of a and b
>NNN	NNN in hexadecimal number format
NNN	NNN in decimal number format
=	Equal
<>	Not equal
>	Greater than
≥	Greater than or equal to

<	Less than
≤	Less than or equal to
+	Addition
-	Subtraction
.OR.	Boolean OR function
.AND.	Boolean AND function
PC + 1 → PC	PC value goes to next word address in the pseudo random sequence (00,01,03,07,0F etc.). The complete sequence is given in Figure 2.
$\overline{X}$	One's complement of X
XXX.YY	Number with 2 digits after the decimal point

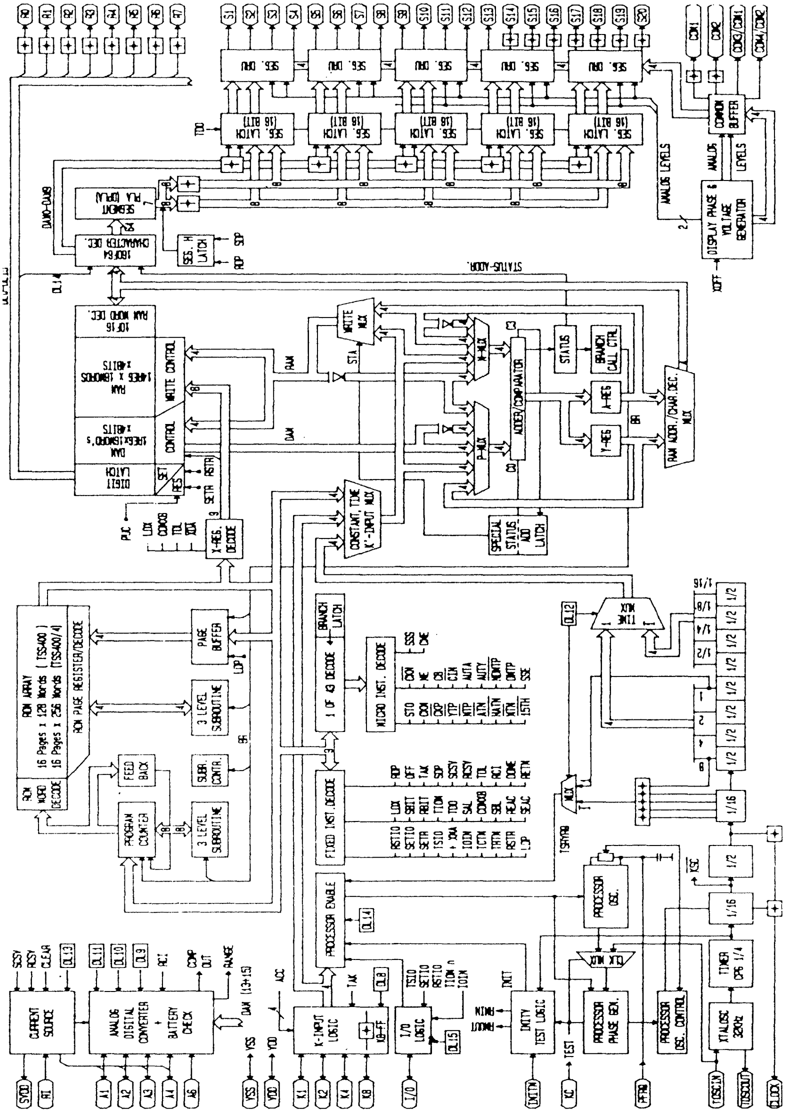


Figure 1: Blockdiagram TSS400(/4)

2 TSS400(/4) Chip Architecture and Operation

2.1 General

The instruction timing is fixed and each instruction requires six oscillator cycles to execute. By initalization, the Program Counter PC is reset to location 00 and the Page Address register PA and the Page Buffer PB are set to 15. Then the Program Counter counts to the next ROM address in a pseudo random sequence. If the PB is not changed, execution continues on the same page. In other words, when the PC reaches the 128th word on a page, execution begins again at PC location 00 on that page (wrap around).

The actual available signals (Kn, Rn, Select Lines etc.) depend on the ADT configuration program. All signals are described wether existent or not in a given configuration.

2.2 ROM Addressing

The ROM has 2048 words or 4096 words in the 4k version of 9 bits each. NMOS transistors are placed to define bit patterns of machine language. The ROM is organized into 16 pages of 128 words or 256 words in the 4k versiojn each (16 · 128 = 2048 words total).

Registers used to address the ROM include the following:

a) Page Address register PA: Contains the page number within the ROM being addressed. The contents of the PA (four bits) are decoded into one of sixteen address lines by the page decoder.

b) Page Buffer register PB: The PB is loaded with a new page address by the LDP instruction. This page address is transferred to the PA when a successful CALL or BRANCH instruction is executed.

c) Program Counter PC: Contains the current location of the word (within the current page) being addressed. The contents of the PC (seven bits length) are decoded by the PC decoder into one of 128/256 address lines selecting one instruction on a page.

The PC steps forward in a pseudo random sequence. This sequence is shown below: (00, 01, 03 .... 40)

→							
00	01	03	07	0F	1F	3F	7F
7E	7D	7B	77	6F	5F	3E	7C
79	73	67	4F	1E	3D	7A	75
6B	57	2E	5C	38	70	61	43
06	0D	1B	37	6E	5D	3A	74
69	53	26	4C	18	31	62	45
0A	15	2B	56	2C	58	30	60
41	02	05	0B	17	2F	5E	3C
78	71	63	47	0E	1D	3B	76
6D	5B	36	6C	59	32	64	49
12	25	4A	14	29	52	24	48
10	21	42	04	09	13	27	4E
1C	39	72	65	4B	16	2D	5A
34	68	51	22	44	08	11	23
46	0C	19	33	66	4D	1A	35
6A	55	2A	54	28	50	20	40

Figure 2: PC Stepping Order

2.3 Branching

All branches are conditional; a status logic path comes from the ALU to designate if a branch instruction should be successfully executed. A successfully executed branch or call is defined to be the case when the branch or call transfers control to an instruction address out of the normal sequence. An unsuccessful branch or call does not affect the normal sequence of the Program Counter. The execution of a branch or call depends on the status:

- If the status equals one, then the branch will be successfully executed. That is, seven bits are transferred from the instruction bus from ROM into the Program Counter. These seven bits are the branch address W which locates the next word on the page to be executed.
- If the status logic is equal to zero, then the branch instruction is unsuccessful. The Program Counter sequences to the next instruction, and then status reverts to a one.

If the contents of the Page Buffer register had been modified previous to the branch instruction, then this instruction is called a long branch instruction, since it may branch anywhere in the ROM (a long branch, the BL directive in the source program, generates two instructions: LDP load Page Buffer and BR branch).

Note: The normal state of the status logic is one. Several instructions can alter this state to a zero; however, the zero state lasts for only one subsequent instruction cycle (which could be during a branch or call), then the status logic will normally revert back to its one state (unless the following instruction resets it to zero again).

2.4 Subroutines

Similar to the BRANCH instruction, the CALL instruction is conditional, too. Three levels of subroutine nesting are allowed and a 4th level CALL does not work properly. In the case of a successful CALL when status logic equals 1:

- the contents of the Page Buffer PB are written to the Page Address PA
- the return address is pushed onto the stack (PSR1 and SR1) (the return address is the address after the call instruction and is called normally PC + 1)
- the address field W of the CALL instruction is written into the Program Counter PC

When a return instruction RETN occurs in a subroutine:

- the Subroutine PC Register SR1 (which holds the address after the last CALL instruction) is written to the PC
- the Subroutine Page Register PSR1 is written to the PA
- the stack is raised one level up

2.5 RAM Addressing

There are 9 RAM banks each containing 16 four-bit words (nibbles) in the RAM's 576 bit matrix of the TSS400. There are 15 RAM banks each containing 16 four-bit words (nibbles) in the RAM's 960 bit matrix of the TSS400(/4). Eight of these registers are normal RAM registers, one of them is a special Direct Access Memory (DAM) register. Special instructions address this DAM without changing the X register. When addressed implicitely (X = 7 or 15) all instructions operate in the same manner on the DAM as when used for the normal RAM banks.

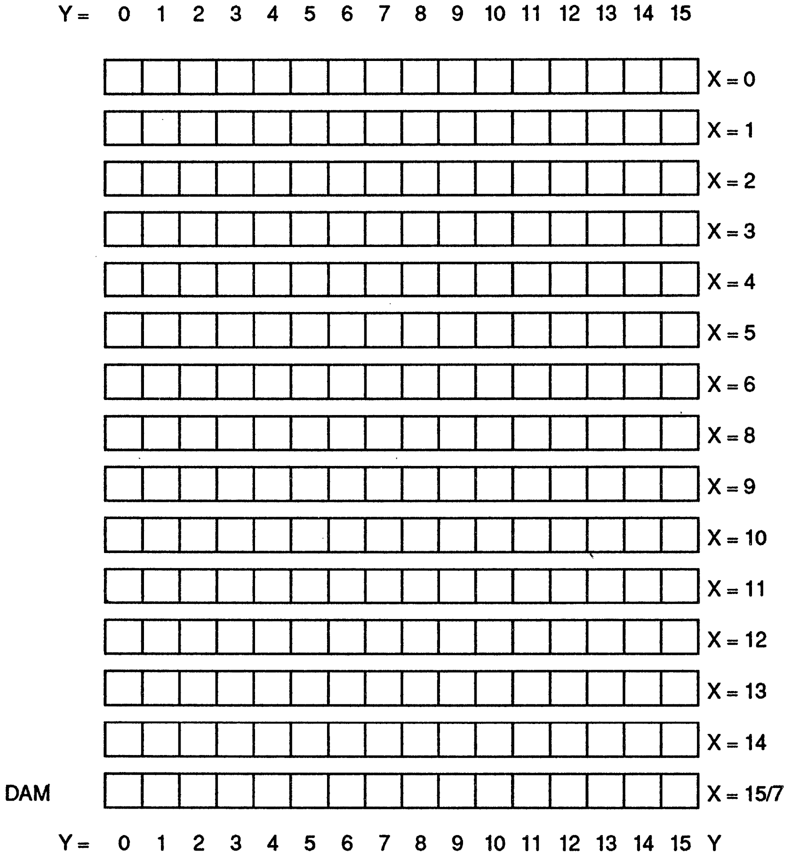


Figure 3: RAM Map TSS400

Usage of a RAM map like shown above is strongly recommended. It makes the RAM usage very clear and eases coding and test.

Three registers are important in RAM adressing:
- the X register addresses each bank with a four-bit address only (1, 2, 3, 4, 7, 9, 10, 11, 12, 15 in the TSS400), the address being decoded by the X decoder
- the Y register identifies the particular nibble in the bank with a four-bit address (0 to 15). The Y register is decoded by the Y decoder
- the DAM Latch decides if the X register is used for the addressing of a register bank (DAM Latch = 0) or if the DAM is addressed independently from the X register contents (DAM Latch = 1)

The DAM Latch will reset upon a LDX instruction and will set and reset on subsequent TDL instructions. When the DAM Latch is reset, the RAM bank addressed by the X register is in use again.

2.6 The Y Register

The Y register has four purposes.
- the Y register addresses the RAM in conjunction with the X register for RAM I/O
- the Y register addresses the Digit Latches for setting and resetting individual latches. Whenever a particular Digit Latch needs to be set or reset, a TCY instruction with the appropriate Y and then a SETR or RSTR instruction is to be executed
- the Y register is used as a working register; for example, a long delay time is desired, the Y register is used as a counter (see subroutine WAIT2MS in example 6.14)
- the Y register addresses the display digit which is to be loaded with the TDO instruction

Example: For setting the R0 output the following program can be used.

```
TCY     0     0 → Y (ADDRESS R0)
SETR          SET R0
```

The following instructions affect the Y register:

CTMDYN	Conditional transfer to memory + DYN
DYN	Decrement Y register, if no borrow one to status
IYC	Increment Y register, if carry one to status
TAMDYN	Transfer accumulator to memory + DYN
TAMIYC	Transfer accumulator to memory + IYC
TAY	Transfer accumulator to Y register
TYA	Transfer Y register to accumulator
TCY	Transfer constant to Y register
TCMIY	Transfer constant to memory. Increment Y register
TMY	Transfer memory to Y register
TTY	Transfer timer to Y register
YNEA	If Y register not equal to accumulator, set status
YNEC	If Y register not equal to constant, set status
ACYY	Add constant to Y register

The ACYY instruction allows any constant (0-15) to be added to the Y register. This is helpful for addressing transfer loops.

2.7 The R Outputs and Digit Latches

The Y register addresses 16 Digit Latches which can be set and reset by the instructions SETR and RSTR. Initialization will reset all Digit Latches to zero. The data of the Digit Latches is retained during Done Mode and Off Mode as long as the supply voltage is maintained.

Two groups of Digit Latches can be distinguished:

- R outputs: The latches which are addressed by Y values zero to seven are connected to these outputs named R0 to R7. All of them are equal in function. They can be used for several purposes:
  - scanning of a keyboard, or switches or programming diodes
  - controlling of relays, lamps, LEDs etc. via buffers
  - switching of gates, multiplexers etc.

- Digit Latches: These latches are addressed by Y values eight to fifteen. Each Digit Latch DLn has a unique function as described below:
  - DL8: Controls the direction of the K-Port:
    DL8 = 0: K-Port is input
    DL8 = 1: K-Port is output
  - DL9 - DL11: Control of the analog multiplexer of the ADC:

DL11	DL10	DL9	Function
0	0	0	A1 to comparator
0	1	0	A2 to comparator
1	0	0	A3 to comparator
1	1	0	A4 to comparator
x	x	1	Battery check

Figure 4: Analog-Digital-Converter Addressing

- DL12: Controls which timer register is connected to the ALU or memory for the timer instructions and functions:
  DL12 = 0: Timer0 (0 - 15 s) is addressed
  DL12 = 1: Timer1 (0 - 15/16 s) is addressed
- DL13: Controls the constant current source of the ADC:
  DL13 = 0: Constant current source is off
  DL13 = 1: Constant current source is on if SV_{DD} is on
- DL14: Is MSB for the OPLA character decoding and selects wake up direction of the K8 input.
  DL14 = 0:
  Leading edge at K8 generates wakeup (LO-HI change)
  Characters 0 to 31 are addressed in the OPLA
  DL14 = 1:
  Trailing edge at K8 generates wakeup (HI-LO change)
  Characters 32 to 63 are addressed in the OPLA
- DL15: Controls wakeup of the I/O Pin:
  DL15 = 0: Wakeup is enabled for I/O Pin
  DL15 = 1: Wakeup is disabled for I/O Pin

2.8 The Accumulator Register

The accumulator is a four-bit register that interacts with the adder, the RAM, the DAM, the Y register, the LCD driver and the K-Ports. The accumulator is the main working register for addition and subtraction. Subtraction is accomplished by complement of two arithmetic. The functions are as follows:
- main register for computations
- selection of OPLA term for display together with status and DL14
- storage register for K-Port (input and output)

2.9 Arithmetic Logic Unit

Arithmetic and logic operations are performed by the Arithmetic Logic Unit (ALU) which is a four-bit adder/comparator and associated logic. The Arithmetic Logic Unit performs logical comparison, addition, subtraction and arithmetic comparison functions on its inputs.

Two input groups are available: P input and N input. These two four-bit parallel inputs (P and N) may be added together or compared logically.

The inputs of the P group are:
- Y register
- constant from instruction
- addressed DAM nibble
- addressed DAM nibble inverted
- addressed RAM nibble

The inputs of the N group are:
- addressed RAM nibble
- constant from instruction
- accumulator
- accumulator inverted

The adder has additional inputs for:
- 15 (-1) for decrementing purposes
- special Status SS
- 1 for incrementing purposes

Addition and subtraction results are stored in either the Y register or the accumulator. Either an arithmetic function or a logical comparison may generate an output to the status logic. If either logical or arithmetic comparison functions are used only the status bit affects the program control, and neither the Y register's nor the accumulator's contents are affected. If a branch or call is attempted when the status is a logic 1 (which is the normal state), the conditional branch or call is executed.

If an instruction calls for a carry output to status and the carry does not occur, then the status will go to a zero level for one instruction cycle. Likewise, if an instruction calls for the logical comparison function and the bits compared are all equal, then the status will go to a zero level for one instruction cycle. If status is a logical zero, then branches and calls are not performed.

For the ORMA and ORMNAA instruction, which allow the logical OR function for the accumulator and the addressed memory, both data sources are connected to the N group inputs.

The ALU has a carry-in feature in which a one is added to the sum of the P and N adder inputs.

2.9.1 The ADD Latch

The ADD Latch, together with the Special Status SS and the BRANCH Latch logic, facilitates the execution of arithmetic with other bases than 16. Normally the used base is 10 (BCD arithmetic). The ADD Latch connects the adder logic optimally for addition if set and for subtraction if reset.

The ADD Latch is affected by the following instructions:

SAL	Set ADD Latch
RETN	Reset ADD Latch

The ADD Latch is reset by the RETN instruction, after initialization and after wakeup from Done Mode and Off Mode.

When the SAL instruction is used and a defined status of the Special Status SS is necessary, the SAL instruction should be used before the REAC or SEAC instruction. This is because the SAL instruction causes an undefined status of the SS.

Example: The subroutine following will add the numbers in M(15,0-5) to the numbers in M(3,0-5) in octal format.

```
OCTALADD  LDX      3         Address LSD of registers
          TCY      5         M(3,5)
          SAL                Set ADD Latch for addition
          REAC               Reset Special Status
ADDLOP    DMEA               Add DAM and RAM → A
          TAMACS   8         A → M(3,Y), A + 8 → A
          CTMDYN             Correction if necessary
          BRANCH   ADDLOP    to octal range 0 - 7
          RETN               Result in M(3,0-5)
```

Note: Only the constant of the TAMACS instructiondecides which numbering base NB is used:
Addition: NB = 16 - TAMACS constant
Subtraction: NB = TAMACS constant

2.9.2 The Branch Latch

The BRANCH Latch is set when it is desired to have the BRANCH instruction decoded not only as a BRANCH, but also as an instruction. The instruction decoded will execute irregardless of status. When the BRANCH Latch is set it will allow the Microinstruction Decode to interprete the BRANCH instruction with the MSB set to zero. This means, that the destination address of the BRANCH must have a ROM address equal to the instruction opcode to be performed together with the BRANCH.

The reason for the BRANCH Latch is the fact that when using arithmetic functions, the program is nearly 100 % inside the add loop of a multiplication or the subtract loop of a division. The Special Arithmetic Instructions of the TSS400(/4) reduce these loops to 4 instructions compared to 11-13 instructions for TMS1000 family members. The usage of the BRANCH Latch gives an additional 25 % speed improvement.

Instructions which cannot be executed in this way are: TKA, TKM, KNEZ, XDA, TBIT, TTM, TTA

The following instructions affect the BRANCH Latch:

SBL	Set BRANCH Latch
RETN	Reset BRANCH Latch

The BRANCH Latch is reset by the RETN instruction, after initialization and after wakeup from Done Mode and Off Mode.

When the BRANCH Latch is set, every BRANCH works as a branch and as an instruction. Use only branches which are defined as instructions when the MSB is zero. If not it is possible that some microinstructions which are activated will cause problems (e.g. transferring adder output to memory). Don't forget to reset the BRANCH Latch by a RETN instruction when finished.

Caution: Be very cautious using this feature. Any shift inside the ROM page will change the instruction which is performed by the BRANCH additional to the branching. Document the used instruction in the comment field and use the ORG Directive for placing the BRANCH instruction. Then a warning will occur if modifications before the ORG directive overflow into the critical loop.

Example: Three instruction BCD Addition and Subtraction.

The ORG operands are computed as follows:

ORG	operand = (PAGE · 128) + CNT
PAGE	Page number (0-15) where instruction is located
CNT	Count from PC = 0 to PC = Opcode

```
PC INSTR. LINE

00 095    2  ADDITION  LDX     10       Add M(15,0-8) to M(10,0-8)
01 041    3            TCY     8
03 0B1    4            SAL              Set ADD Latch for Addition
07 0B3    5            SBL              Set BRANCH Latch for speed
0F 0B4    6            REAC             SS = 0
1F 110    7            BR      ADDLOP   Includes DMEA (010) instr.
          8  *
```

```
3F 095    9  SUBTRACT  LDX      10       Subtract M(15,0-8)
7F 041   10            TCY      8        from M(10,0-8)
7E 0B3   11            SBL               Set BRANCH Latch for speed
7D 0B5   12            SEAC              SS = 1  ADD Latch not set
7B 113   13            BR       SUBLOP   Includes NDMEA (013) instr.
         14  *
         15            ORG      88       Place ADDLOP to PC = 10
10 0D6   16  ADDLOP    TAMACS   6        A -> M(X,Y)
21 018   17            CTMDYN            Correction if necessary
42 110   18            BRANCH   ADDLOP   Includes DMEA (010) instr.
04 012   19            CCLA              Carry from MSD -> A
09 0BF   20            RETN              Reset BRANCH + ADD Latch
         21  *
         22            ORG      93       Place SUBLOP to PC = 13
13 0D5   23  SUBLOP    TAMACS   10       A -> M(X,Y)
27 018   24            CTMDYN            Correction if necessary
4E 113   25            BRANCH   SUBLOP   Includes NDMEA (013) instr.
1C 012   26            CCLA              Carry from MSD -> A
39 0BF   27            RETN              Reset BRANCH + ADD Latch
```

2.10 Status and Special Status

2.10.1 Status Logic

There are several instructions that affect status logic, either setting it to one or resetting it to zero. In turn the status logic will permit the successful execution of a BRANCH or CALL instruction if status is a one, or prevent sucessful execution of these instructions if status is zero. Status logic will remain at a zero level only for the following instruction cycle and then automatically be set to the normal one status by begin of the next instruction.

There are two microinstructions (NE and C8) that are used to affect status. If the C8 microinstruction is used and a carry occurs in the addition of two four-bit nibbles, the carry goes from the MSB of the sum to status, setting the status logic to a one. If no carry occurs, status logic is reset to a zero for one instruction cycle. In a logic compare instruction using the microinstruction NE, status is set to one if the four-bit nibbles at the N and P adder/comparator inputs are not equal, conversely, status logic is reset to zero if the inputs are equal.

The status can be set or reset without any other side effects by the following instructions:

ACACC	0	Reset status unconditionally
ACYY	0	Set status unconditionally

2.10.2 Special Status

The Special Status SS or End-Around-Carry is used by the Special Arithmetic Instructions only. The BRANCH and CALL instructions do not depend on the special status; it is an arithmetic status only. All of the instructions mentioned below (except CTMDYN) unconditionally set the status.

The following instructions depend on or modify the Special Status:

SEAC	sets the SS
REAC	resets the SS
DMEA	adds the SS to the result
NDMEA	adds the SS to the result
DNAA	adds the SS to the result
TAMACS	sets or resets the SS conditionally
CTMDYN	transfers accumulator to memory depending on SS
CCLA	transfers the SS to the accumulator
SAL	switches ALU to addition (to subtraction if not set)

The TAMACS instruction which sets or resets the SS conditionally uses the carry out of each instruction which affluences the carry, not only out of the instructions named above. This means that all arithmetic instructions (including the instruction ACYY which always sets the status) can be used together with the TAMACS instruction. See example 6.10 where the AMAAC instruction is used together with the Special Arithmetic Instructions. The ALU always stores the carry of the previous instruction, too.

2.11 The K-Port

The K-Port is a four-bit I/O port. Its direction is controlled by DL8 while the output data is stored in the four-bit K-Port Latch.

DL8 = 0: K-Port is switched to input direction
DL8 = 1: K-Port is switched to output direction. The K-Port Latch information appears at the K-Pins.

Because initialization resets DL8, the K-Port is switched to input direction when initialization occurs.

The following instructions affect the K-Port:

TKA	Transfer K-Port to accumulator
TKM	Transfer K-Port to addressed memory
KNEZ	If K-Port input not equal to zero, set status Reset status if K-Port input is zero
TAK	Transfer accumulator to K-Port Latch

The TAK instruction latches the four-bit accumulator into the K-Port Latch. If DL8 is set, the K-Port Latch is output to the K-Pins. K8 is MSB, K1 is LSB. The contents of the K-Port Latch are not affected if DL8 is switched to input direction.

Example: The example outputs a memory value to the K-Port

```
TMA              M(X,Y) → A
TAK              A → K PORT LATCH
TCY      DL8     SWITCH K PORT TO OUTPUT
SETR             K PORT LATCH → K PORT
```

The K-Port input instructions TKM, TKA, KNEZ read the K-Port information via SCHMITT-triggers into the ALU or memory. This means that depending on the state of DL8 the input is read in or the information coming from the K-Port Latch.

The K-Port will wakeup the TSS400(/4) from Done Mode and Off Mode under certain circumstances. See Figure 5 for the conditions of wakeup at the K-Port.

2.11.1 K8 Wakeup Conditions

K8 differs from the other K-Pins in that the wakeup conditions depend on DL14 (See also Figure 5):

DL14 = 0: K8 transition from 0 to 1 will wake up the CPU
DL14 = 1: K8 transition from 1 to 0 will wake up the CPU

This feature is for applications with relatively high input frequencies which have to be handled. Before entering the Done Mode or Off Mode, the software checks the state of the K8 input and sets DL14 accordingly for a safe wakeup.

```
        TCY      DL8       SWITCH K PORT TO INPUT
        RSTR
        TCY      DL14      ADDRESS DL14
        TKA                READ IN K PORT
        TBITA    3         IS K8 HI ?
        BRANCH   L$080
        RSTR               K8 IS LO, RESET DL14 FOR WAKEUP
        DONE               BY LO-HI CHANGE
*
L$080   SETR               K8 IS HI, SET DL14 FOR WAKEUP
        DONE               BY HI-LO CHANGE
```

Changes of K8 occuring between the TKA instruction and the DONE instruction will also wake up the ALU, because these changes are stored in a 5 instruction shiftregister. It wakes up if the current wakeup direction is addressed. Due to this shiftregister, no unnecessary DL14 changes should be made within 6 instructions before the DONE instruction. The following example would wakeup erroneously the ALU each time after the DONE instruction in an endless loop: Dependent on the state of K8, the conditions for wakeup inside the stored 5 instructions can be met.

If K8 is 0: The SETR instruction simulates a HI-LO change
If K8 is 1: Correct function

```
        TCY      DL8       SWITCH K PORT TO INPUT
        RSTR
        TCY      DL14      ADDRESS DL14. PRESET IT FOR
        SETR               HI-LO CHANGE (WRONG !!)
        TKA                READ IN K PORT
        TBITA    3         IS K8 HI ?
        BRANCH   L$080     YES, DL14 YET RIGHT
        RSTR               K8 IS LO, RESET DL14 FOR WAKEUP
L$080   DONE
```

2.11.2 K8 Input Dividing by Flip-Flop

Fast input transitions are very hard to count, especially if the pulses are of short duration. To compete with this situation a hardware option can be selected which divides the K8 input with a flip flop. Each input pulse at K8 changes the output of the flip flop and stores this information until the next K8 input pulse occurs. This flip flop is reset by initialization or powerup. The first LO-HI change at the K8 input delivers a logical 1 to the ALU. The current state of the K8 pin is not available to the ALU. The wake up information, which depends on DL14, is triggered by the K8 pin directly and is not affected by this option.

The K8 flip flop is controlled by the .OPTION K8FF. See 2.18.4.

2.11.3 K-Port Options

Each K-Pin can have a pull-down resistor individually by defining it with the hardware option .OPTION K1K2K4K8IO. See 2.18.5 for examples. It is strongly recommended for all K-Pins which are not connected to external resistors or gate outputs to have internal pulldown resistors. This avoids floating input levels and EMI influences.

The output configuration of each K-Pin is selectable by the .OPTION K1K2K4K8IO, too. Two possibilities exist:
- K-Pin is configured with push pull output
- K-Pin is configured with open source output

Pulldown resistor and output configuration can be chosen independently for each K-Pin. See 2.18.5 for examples.

2.12 The I/O Pin

The I/O Pin is a one-bit bidirectional port which is controlled by its own instructions. The I/O Pin can be used for a two line bus or other purposes:

- additional input or output to K-Port and R outputs
- connection to a host computer

- connection to a calibration unit
- read in and read out of measurement results

The I/O Pin is controlled by two one-bit registers:
- The I/O Data Latch which holds the output information. If set to one, the output info is nearly V_{DD}, if reset to zero the output info is nearly V_{SS}.
- The I/O Control Latch which controls the direction of the I/O Pin. This latch is reset to input direction by initialization.

Both I/O latches hold their information during Done Mode and Off Mode. The I/O Pin output information settles in the middle of an instruction.

The instructions which affect the I/O Pin are:

SETIO set I/O Data Latch to 1 and set I/O Control Latch to output direction

RSTIO reset I/O Data Latch to 0 and set I/O Control Latch to output direction

TSIO transfer status to I/O Data Latch and set I/O Control Latch to output direction

IOIN reset I/O Control Latch to input direction

TIOM transfer I/O pin information to addressed memory (I/O Pin is not switched to input direction !)

The I/O Pin can have an internal pulldown resistor if defined by the hardware option .OPTION K1K2K4K8IO. See 2.18.5.

Example: The following code output a HI signal at the I/O Pin, three instructions in length which is a trigger signal for the host and input then one bit of information.

```
READ      SETIO                   SET I/O PIN HI
          LDX        4            FOR 3 INSTRUCTIONS
          TCY        0            TO TRIGGER HOST
          RSTIO
          IOIN                    SWITCH TO INPUT DIR.
          TIOM       3            READ INFO TO M(4,0,3)
```

2.13 The Timer

The TSS400(/4) has an eight bit timer register which is fed by a frequency derived from the oscillator after dividing by 2048 (16 Hz for 32768 Hz).

The oscillator circuitry is selectable (see hardware option .OPTION TIMOSC). Two posibilities exist:
- Quartz oscillator 10 to 100 kHz (nominal 32768 Hz)
- RC oscillator

The quartz oscillator circuitry is intended for applications which need an accurate time base. The RC oscillator circuitry can be used when no time base is necessary and very fast startup after powerup or release from Off Mode is essential (the internal powerup logic delays the program start until 6 to 8 clock cycles came from the oscillator. This can last up to 0.5 seconds with a quartz oscillator. The RC oscillator starts immediately).

The timer is divided into two nibbles Timer0 and Timer1, which can be read into the accumulator, the addressed memory or the Y register. The frequency information in the remaining timer description is always related to an input frequency of 32768 Hz.

Timer0: Fed by 1 Hz coming out of the MSB of Timer1.
Addressed if DL12 is reset. Counts from 0 to 15 seconds.

Timer1: Fed by 16 Hz coming out of the divider chain of the timer oscillator.
Addressed if DL12 is set. Counts from 0 to 15/16 seconds.

For intervals greater than 15 seconds the timing information has to be enlarged by software using the RAM. See example below. The timer can be used for several purposes:
- time base for software timers
- measurement of time intervals
- keyboard debounce
- wait routines
- wakeup from Done Mode: If DL12 is reset, wakeup will occur each second. If DL12 is set, wakeup will occur with the frequency defined by hardware option .OPTION WAKEUP (see 2.18.8) Possible frequencies are 8, 16, 32, 64 and 128 Hz.

The following instructions handle the timer information:

TTA	transfer addressed timer nibble to accumulator
TTM	transfer addressed timer nibble to M(X,Y)
TTY	transfer addressed timer nibble to Y register
TNEA	if addressed timer nibble is not equal to accumulator set status to one; reset status if both are equal;
TNEM	if addressed timer nibble is not equal to M(X,Y) set status to one; reset status if both are equal;

The MOS oscillator and the timer oscillator are completely independent from each other. This means that the timer information can be in an intermediate state when read out by the instructions TTY, TTA or TTM. This would lead to false readings. The TNEA and TNEM instructions are used to ensure that the read timer information is stable. If the read timer values differs, another double read is made until both are equal. See example below.

When DL12 is reset the value of Timer1 is 0 after wakeup is caused by the timer. This can be used for assuring that the timer woke up the CPU. If Timer1 doesn't contain 0, the wakeup was caused by the K-Port, the I/O Pin or noise.

When DL12 is set and the used wakeup frequency is 32 Hz and higher it is not as easy as above to decide where the wakeup came from. The K-Port and the I/O pin have to be checked if they woke up the CPU before the time counter is incremented. If high time accuracy is needed, it is advised to synchronize the time with the readable 16 Hz Timer1.

Example: The following program reads the Timer0 and builds the difference to the last read value. This difference is used for incrementing the clock routine which is shown in 6.10. Using the difference of readings assures that neither noise nor long computations (up to 15 seconds) can influence timing accuracy. It can not be recommended to increment a timer simply after wakeup!

```
         TCY        DL12        ADDRESS TIMER0
         RSTR
         TCY        Y           M(X,Y) OLD VALUE
L$100    TTA                    TIMER0 -> A
         TNEA                   TIMER0 = A ?
         BRANCH     L$100       READ ONCE MORE IF NOT
         XMA                    OLD <-> NEW
         SAMAN                  NEW - OLD -> A
         CALLL      CLOCK       ADD DIFF. TO CLOCK
```

By using the hardware option .OPTION CLOCK and a 32768 Hz quartz it is possible to have either 256 Hz or 1024 Hz at a pin named CLOCK. This frequency is output in the Done Mode, too. If quartzes with other frequencies or an RC oscillator are used, the output frequency is 1/128 resp. 1/32 of the used frequency. After execution of the OFF instruction the state of the CLOCK output is not defined, it can be HI or LO. If a certain state is necessary, the CLOCK output has to be supervised via a K-input and the OFF instruction executed when the wished state is reached.

This CLOCK output can be used for the timing of external devices or for the improvement of the time measurement accuracy of the TSS400(/4). For this purpose, the CLOCK output is connected to a K-input via a gate controlled by an R output to allow the Done Mode without continuous wake up by the CLOCK frequency. During the critical time measurement the R output is switched on and the software can count the CLOCK pulses for better resolution.

2.14 The Current Saving Modes

It is not necessary for the TSS400(/4) to stay on at all times. The processor can be programmed to come on only if it is needed. This can be for updating the timekeeping or for performing a software function. The advantage is the vast amount of current saving by using these features of the TSS400(/4). Two different modes of current saving are possible: The Done Mode and the Off Mode. The following chapter will describe the common elements of both modes:

Wakeup from initialization: see Figure 11 for conditions.

Wakeup from K-Port or I/O Pin: See Figure 5. Depending on the Digit Latches DL14 and DL15 certain conditions must be met that wakeup can occur. Only if the conditions described under "before wakeup" are met, the conditions described under "wakeup condition" will wakeup the CPU from Done Mode or Off Mode.

For example if DL14 = DL15 = 0: Only if all K-Ports and the I/O Pin are low, then the LO-HI change of anyone of the inputs will wakeup the TSS400(/4).

DL15	DL14	I/O Pin	K8	K4	K2	K1	Comment
0	0	0	0	0	0	0	before wakeup
		1 or	1 or	1 or	1 or	1	wakeup condition
0	1	0	1	0	0	0	before wakeup
		1 or	0 or	1 or	1 or	1	wakeup condition
1	0	X	0	0	0	0	before wakeup
		X	1 or	1 or	1 or	1	wakeup condition
1	1	X	1	0	0	0	before wakeup
		X	0 or	1 or	1 or	1	wakeup condition

Figure 5: Wakeup Conditions for external Events

The processor registers wake up with the following contents:

Register	Done Mode	Off Mode
Program Counter PC	>7F	>7F
Page Address PA	15	15
Page Buffer PB	15	15
Status S	undefined	undefined
Special Status SS	undefined	undefined
Accumulator A	undefined	undefined
Y register Y	undefined	undefined
X register X	undefined	undefined
RAM Contents	unchanged	unchanged
DAM Latch	undefined	undefined
ADD Latch	reset to 0	reset to 0
BRANCH Latch	reset to 0	reset to 0
Decimal Point DP	reset to 0	reset to 0
Digit Latches DLn	unchanged	unchanged
K-Port Latch	unchanged	unchanged
I/O Port Data Latch	unchanged	unchanged
I/O Port Control Latch	unchanged	unchanged
Timer0 and Timer1	actual time	undefined
ADC voltage SV_{DD}	switched off	switched off
LCD segment latches	unchanged	unchanged
Subroutine Stack	level 0	level 0

Figure 6: Wakeup Contents of internal Registers

Note: Due to the undefined state of the status a BRANCH or CALL instruction located at page 15 PC 7F (wakeup address) will not be executed safely. It is strongly recommended to use an instruction which is executed independently from the status at the wakeup address (LDX e.g.)

When in Active Mode, a delay line is active which stores all events that could wake up the CPU in Off Mode or Done Mode up to 5 instruction cycles. This ensures that no event is missed if the DONE instruction follows within 5 instructions after the last software check. This delay is not active when in Done Mode or Off Mode. The CPU wakes up immediately in these cases.

Example: To ensure safe wake up in situations described as above the sequence before the DONE instruction should look like:

```
TKA           LAST CHECK FOR CHANGES
...
...           3 INSTRUCTIONS MAX.
...
DONE          OR OFF INSTRUCTION
```

2.14.1 The Done Mode

During the Done Mode the CPU is switched off completely. Only the timer, the RAM banks, the LCD driver, the Digit Latches and certain other registers are active. See Figure 6 for more information concerning data retention during the Done Mode. The other registers do not maintain information. Wakeup from this mode can come from the timer, the K-Port, the I/O Pin or initialization.

Wakeup from timer: Dependent on DL12 the TSS400(/4) will be woken up with 1 Hz (DL12 = 0) or with a frequency ranging from 8 Hz to 128 Hz selectable by the hardware option .OPTION WAKEUP-FREQUENCY (DL12 = 1). The wakeup from the timer occurs always, independent of the state of the K-Port or I/O pin. When woken up by the timer, there must be a minimum number of instructions N until the DONE instruction can be used.

$$N \geq \frac{4 \cdot \text{Max. processor frequency}}{6 \cdot \text{Min. timer frequency}}$$

The Done Mode is entered by execution of the DONE instruction. The ADC voltage SV_{DD} is switched off by entering the Done Mode. The frequency output at the CLOCK pin continues during the Done Mode.

2.14.2 The Off Mode

During the Off Mode the CPU is switched off completely. Only the RAM banks and Digit Latches and certain other registers are active. See Figure 6 for more information concerning data retention during the Off Mode. Wakeup from the Off Mode can come only from initialization, the K-Port or the I/O Pin. See Figure 5 for wakeup conditions. The current consumption is further reduced compared to the Done Mode.

The Off Mode is entered by execution of the instruction OFF. The ADC voltage SV_{DD} is switched off by entering the OFF Mode. The CLOCK output remains the current state (HI or LO) after execution of the OFF instruction.

2.15 The Analog Digital Converter

The TSS400(/4) contains an Analog Digital Converter with a resolution of 12 bits. Up to 4 analog inputs are possible which are named A1 to A4. The input selection is made by DL9, DL10 and DL11:

DL11	DL10	DL9	Function
0	0	0	A1 to comparator
0	1	0	A2 to comparator
1	0	0	A3 to comparator
1	1	0	A4 to comparator
x	x	1	Battery check

Figure 7: Analog-Digital-Converter Addressing

The ADC does not measure absolute voltages but the ratio of the input voltage to the supply voltage. So the measurement of sensors is independent of the supply voltage. Absolute measurements are possible if SV_{DD} is held constant, which means stable V_{DD} and very small loading at SV_{DD}.

The following instructions control the A/D converter:

SCSV Switches on the SV_{DD} of the ADC. Switches comparator inputs to noninverted.

RCSV Switches off the SV_{DD} of the ADCT.

CTM Transfers output of comparator to addressed memory bit and sets next lower bit to 1.

TRTM Sets addressed memory bit to 1 if input voltage is out of ADC range (result 000 or >FFF). Resets bit otherwise.

RCI Switches comparator inputs to inverted.

The Analog Digital Conversion can be done by a simple software program using successive approximation. The memory nibbles M(15,13-15) are connected to a Digital Analog Converter (DAC) where M(15,13,3) is the MSB and M(15,15,0) is the LSB. Starting with the midpoint of the range (>800) the input voltage is compared to the output voltage of the DAC and the comparator output written to the addressed memory bit. Additional the next lower memory bit is set for the next comparison (no bit is set if M(X,Y,0) is addressed). With 12 comparisons a 12 bit result is built.

The basic principle of the successive approximaton routine is shown in the next example:

```
          LDX      15        LOAD >800 INTO
          TCY      13        M(15,13-15)
          TCMIY    8         TO HAVE MIDPOINT OF RANGE
          TCMIY    0
          TCMIY    0
          TCY      13        START WITH MSD M(15,13)
ADCLOOP   SBIT     3         SET MSB OF NIBBLE (WHICH IS NOT
          TCTM     3         SET BY TCTM 0 BEFORE)
          TCTM     2         MAKE 4 COMPARES PER NIBBLE
          TCTM     1
          TCTM     0
          IYC                NEXT LOWER NIBBLE
          YNEC     0         LSD BUILT YET ?
          BRANCH   ADCLOOP   IF NOT, PROCEED
          ...                12 BIT RESULT IN M(15,13-15)
```

To get the full accuracy of the ADC, slight modifications of the above idealistic example are recommended:

- A delay should be inserted between the modification of a DAC bit and the comparison with the input voltage. This delay should depend on the bit in use: When building the MSB of the ADC value a larger voltage swing occurs when the DAC bit is changed as compared with the change of the LSB. See 2.15.5 for details.
- For compensation of the offset error of the comparator, one more A/D conversion should be made with inverted comparator inputs. When added together with the result of the noninverted A/D conversion result, the offset error is eliminated.
- After each A/D conversion (inverted and noninverted) a test should be made if the input voltage was inside the range of the ADC. This is simply done by using the TRTM instruction which sets the addressed memory bit if the input voltage was outside of the range (result is 000 for underflow or >FFF for overflow) and resets the addressed memory bit if the input voltage was inside the range.

The above mentioned three improvements are included in the examples shown in 6.7. See there for explanations.

2.15.1 Measurement Ranges and Conversion Formulas

Four different measurement ranges are selectable by the hardware option .OPTION ADC-RANGE. The chosen range is the same for all inputs A1 to A4. The nominal properties of the 4 ranges are listed below:

N A/D conversion result for single measurement
Vmin Vin/SV_{DD} for N = 1
Vmax Vin/SV_{DD} for N = 4094 (>FFF - 1)
Resolution Delta in µV/SV_{DD} for 1 bit result difference

ADC RANGE	Vmin	Vmax	Resolution
LARGE	0.101309	0.494607	96.1
SMALL	0.231314	0.407511	43.0
MEDIUM-LOW	0.100369	0.402501	73.8
MEDIUM-HIGH	0.232683	0.500413	65.4

Figure 8: Analog-Digital-Converter Ranges

The nominal conversion equations for single measurements are:

LARGE	$Vin/SV_{DD} = 0.101213203 + 0.000096090233 \cdot N$ $N = 10406.88496 \cdot Vin/SV_{DD} - 1053.314159$
SMALL	$Vin/SV_{DD} = 0.231271438 + 0.000043048228 \cdot N$ $N = 23229.76 \cdot Vin/SV_{DD} - 5372.38$
MEDIUM-LOW	$Vin/SV_{DD} = 0.100294841 + 0.000073816955 \cdot N$ $N = 13547.02312 \cdot Vin/SV_{DD} - 1358.696532$
MEDIUM-HIGH	$Vin/SV_{DD} = 0.232617939 + 0.000065411591 \cdot N$ $N = 15287.81046 \cdot Vin/SV_{DD} - 3556.218954$

If, as recommended, even numbered measurement counts M are used and the results are added together, the above mentioned conversion equations have to be modified as follows:

$Vin/SV_{DD} = A + B \cdot N$	equation for single measurement
$Vin/SV_{DD} = A + (B \cdot N)/M$	equation for M measurements added
$N = E \cdot Vin/SV_{DD} - F$	equation for single measurement
$N = (E \cdot Vin/SV_{DD} - F) \cdot M$	equation for M measurements added

Note: The useable range of the Analog Digital Converter starts at values greater than 1 and ends at values less than >FFE due to the offset of the comparator. The useable range depends on the V_{DD} and the chosen hardware ADC range. See the specification for actual values.

2.15.2 Correction of Tolerances by Software

The conversion equations mentioned in 2.15.1 show nominal values which in reality differ due to tolerances of Vmin and Vmax. The sensors connected to the ADC have tolerances too, so calibration of the complete system is necessary by hardware (potentiometer, resistors) or by software. The later one is possible if the TSS400(/4) is connected always to a battery. Correction variables once written into the RAM via the I/O Pin are used after the A/D conversion and nominal value computation for correcting the result.

Example: The subroutine TEMPCOMP (see 6.11.2) computes the nominal temperature T_{nom} out of the ADC value. RAM locations contain correction values C and D for adjusting the temperature value to the corrected temperature Treal:

$T_{nom} = N \cdot 0.1363 - 6.696$ TEMPCOMP constants
$T_{real} = Tnom \cdot C + D$ Correction formula
C = Slope correction
D = Offset correction

```
CALLL     TEMPCOMP  HEX ADC -> FLAC BCD         XXX.YY
CALLL     CLRREGB   +0 -> REGB
CALLL     C->RB     COPY C -> REGB              X.YYYY
CALLL     MULTIPL   C x TNOM -> FLAC        XXX.YYYYYY
CALLL     DIV10E4   DIVIDE/ROUND BY 10000       XXX.YY
CALLL     D->RB     COPY D -> REGB              +-X.YY
CALLL     ADDITION  TNOM x C + D -> FLAC        XXX.YY
```

2.15.3 Battery Check

When DL9 is set, the analog inputs A1 to A4 are switched off the ADC and the battery check connection is made: A reference voltage is connected to the ADC input and can be measured. Due to the ratiometric measurement principle of the ADC, the measured digital value is an indication of the supply voltage of the TSS400(/4). The measured value is inversly proportional to the supply voltage V_{DD}. To get the reference for later battery tests a measurement is made with $V_{DD} = V_{DDmin}$. The result is stored in the RAM.

Two possibilities are shown for the battery test:

- A complete measurement is made and the result compared to the stored value measured with $V_{DD} = V_{DDmin}$. If the measured value exceeds the stored one, then $V_{DD} < V_{DDmin}$ and a "Battery low" indication can be given by software. See example 6.12.1
- The stored RAM value for $V_{DD} = V_{DDmin}$ (measured only with inverted comparator inputs) is transferred to the ADC hardware buffer and compared with one instruction against the reference voltage. If the comparator output is 1, the warning level is reached and a "Battery low" indication can be given by software. See example 6.12.2

2.15.4 The Current Source

A switchable current source is available for supplying special sensors. DL13 switches the current source which is simply programmable by an external resistor:

- DL13 = 0: current source is off
- DL13 = 1: current source is on if SV_{DD} is on

When switched on, the current source feeds the constant current out of the addressed analog input An. The voltage generated at the external sensor is measured with the same input.

The current I programmed by the external resistor Rext is given by the following equation:

$$I = V_{rext}/R_{ext}$$

V_{rext} depends sligthly on the chosen ADC range and is approximatively $0.24 \cdot SV_{DD}$.

The current range for each ADC input is as follows:

$$0.15 \text{ to } 2.4 \text{ mA} \cdot SV_{DD}/V$$

This current range is sufficient to drive PT100 sensors, too. (Platinum temperature sensors with a nominal resistance of 100 Ohm)

2.15.5 Timing Considerations for the Measurement

As mentioned earlier, a delay should be inserted between the modification of a DAC bit and the comparison with the input voltage. This delay depends on the bit position of the DAC.

Figure 9 shows the number of delay instructions for each ADC-Bit. The TCTM instruction itself counts too.

ADC-Bit	delay instr.
MSB	5
MSB-1	5
MSB-2	5
MSB-3	5
MSB-4	4
MSB-5	4
MSB-6	4
MSB-7	4
MSB-8	3
MSB-9	3
MSB-10	3
LSB	3

Figure 9: Number of Instructions before ADC-Comparisons

2.16 The LCD Driver

The TSS400(/4) contains a complex LCD driver circuitry which can be modified widely for adapting the TSS400(/4) to the external hardware. The hardware steps the display information has to go through from the software instruction to the common/select line drivers are as follows:

- The software addresses the digit with the Y register and sets DL14, status and accumulator according to the character to be output. Then the TDO instruction is executed which starts the output of the display information.
- The OPLA address defined by DL14, status and accumulator is passed to a decoder, which activates 1 out of 64 output lines.

- The activated output line addresses the segment PLA which outputs the according 7 segments A to G and, dependent on the DP Latch, segment H.
- The segment information out of the segment PLA is transferred via 8 segment lines to the common/select programming area. (This programming area defines which common/select pair activates which segment). The output of the programming area is the segment information adapted to the used LCD display. (See .OPTION SELECT-LINES)
- The decoded ten Y register lines (0 to 9) are transferred to the Y programming area. (This programming area defines which Y register contents activate which digit). The output of the Y programming area addresses the segment latches. (See .OPTION SELECT-LINES)
- The adapted segment information is stored in the 8 segment latches of the addressed digit. Each of these segment latches controls a fixed common/select pair.
- The information of the segment latches is output automatically by the common and select drivers.

From a software point of view the LCD driver looks very simple:

- No timing problems exist with multiplexing for getting a quiet, stable display. The LCD driver hardware outputs the loaded display information automatically, without any software burden, during Active Mode and Done Mode.
- The software only has to decide which segment information (selected by DL14, status and accumulator) has to be displayed at which digit (addressed by Y register).

Steps for defining the display:

- decision if 2 common or 4 common is needed
- definition of Y address for each digit
- definition of select line and common line for each segment
- definition of displayed segments for each DL14, status and accumulator combination

These 4 definition steps are discussed in detail in the following paragraphs.

2.16.1 Multiplexing Scheme

Two multiplexing schemes can be defined for the TSS400(/4) hardware:
- 2 common multiplexing
- 4 common multiplexing

2 common offers better readability, 4 common offers more digits (10 instead of 5 for 2 common). No difference exists for the software between the two multiplexing schemes, the addressing of the (possible) digits is exactly the same for both of them.

The definition is made with the hardware option .OPTION COMMON. See hardware options 2.18.11 for more information.

2.16.2 Digit Addressing

The TSS400(/4) hardware contains a programmable decoder for the addressing of the digits from the Y register. For each digit one (and only one) Y register value can be defined which addresses it. Normally a standard definition is used (see .OPTION SELECT-LINES), but if necessary, this definition is changeable to existing needs. The following example shows the inverse accordance between Y register and select lines than the standard definition defines. If RAM nibble and related digit share the same Y address, additional TCY instructions can be ommitted during output of the display information.

```
.OPTION SELECT-LINES                  ; 2 common
LOAD  S1+S2     BY  Y4                ; Load digit by Y = 4
LOAD  S3+S4     BY  Y4
LOAD  S5+S6     BY  Y3
LOAD  S7+S8     BY  Y3
LOAD  S9+S10    BY  Y2
LOAD  S11+S12   BY  Y2
LOAD  S13+S14   BY  Y1
LOAD  S15+S16   BY  Y1
LOAD  S17+S18   BY  Y0
LOAD  S19+S20   BY  Y0
```

2.16.3 Segment Addressing

For each segment the common line and select line can be defined which activate it. This omits the tedious segment transformation task which is necessary if the used display differs from the standard definition. The common and select line for each segment can be defined with the second part of the hardware option .OPTION SELECT-LINES. The following example shows the inverse select line definition than the standard definition has:

```
!                    COMMON  3 4

FOR  MUX2  AND  S4N-3  LOAD  C D      ; S1,  S5,  S9,   S13,  S17
FOR  MUX2  AND  S4N-2  LOAD  G E      ; S2,  S6,  S10,  S14,  S18
FOR  MUX2  AND  S4N-1  LOAD  A F      ; S3,  S7,  S11,  S15,  S19
FOR  MUX2  AND  S4N    LOAD  B H      ; S4,  S8,  S12,  S16,  S20
```

2.16.4 The Output PLA

The TSS400(/4) LCD driver contains a gate level PLA with 64 x 7 bits. The 7 bits represent the segment information A to G for 64 definable combinations (the segment H info is independent from this OPLA and depends only on the SDP and RDP instructions). This OUTPUT PLA is used to simplify the output of the displayed characters e.g. if a "4" is to be output, the accumulator contains 4 and is written to the segment latches via the OPLA which outputs B,C,F,G for this input code. (Without the OPLA feature the segment representation of 4 (B, C, F, G) has to be built and output.

The addressing of this 64 segment combinations is made by DL14 (MSB), status, and accumulator (4 LSBs).

The definition of this OPLA is made by .OPTION SEGMENT-PLA. See 2.18.13 for this option.

Characters 00 to 15 are addressed by DL14 = 0, status = 0
Characters 16 to 31 are addressed by DL14 = 0, status = 1
Characters 32 to 47 are addressed by DL14 = 1, status = 0
Characters 48 to 63 are addressed by DL14 = 1, status = 1

Some hints:

- If 32 OPLA characters are sufficient, the definitions of the characters 32 to 63 should be made equal to the ones for the characters 00 to 31. This gives independency from DL14.
- Note that DL14 is used with the K8 edge detection too. This can lead to conflicts if display information is output while DL14 has to be stable.
- When defining the OPLA the most used characters (normally the numbers 0 - 9) should be placed from character 16 to 31. (defined for status = 1, the normal state of the status). Binary output for flags can be placed into characters 00 to 15.

Examples: For each character block a control sequence is given.

```
* SEQUENCE OUTPUTS DEFINED CHARACTERS 00 TO 15
*
          TMA                 M(X,Y) -> A
          TCY        DL14     RESET DL14
          RSTR
          ACACC      0        STATUS = 0
          TDO                 OUTPUT CHARACTER 00 TO 15


* SEQUENCE OUTPUTS DEFINED CHARACTERS 16 TO 31
*
          TMA                 M(X,Y) -> A
          TCY        DL14     RESET DL14
          RSTR                STATUS = 1
          TDO                 OUTPUT CHARACTER 16 TO 31


* SEQUENCE OUTPUTS DEFINED CHARACTERS 32 TO 47
*
          TMA                 M(X,Y) -> A
          TCY        DL14     SET DL14
          SETR
          ACACC      0        STATUS = 0
          TDO                 OUTPUT CHARACTER 32 TO 47


* SEQUENCE OUTPUTS DEFINED CHARACTERS 48 TO 63
*
```

```
TMA                    M(X,Y) -> A
TCY         DL14       SET DL14
SETR                   STATUS = 1
TDO                    OUTPUT CHARACTER 48 TO 63
```

The character number in the OPLA definition is the decimal representation of DL14, status and accumulator:

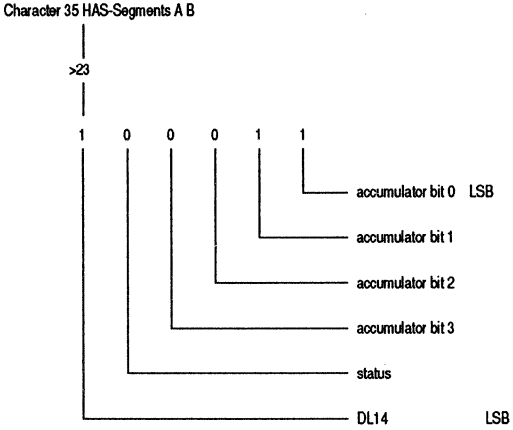


Figure 10: Character Representation in OUTPUT OPLA

Due to the free choice of a segments common/select accordance by the .OPTION SELECT-LINES (See 2.18.12) the OPLA can be defined using the standard segment notation. See Figure 12 for this standard notation.

2.17 Initialization and Power-Up

Initialization can have two hardware reasons:

- Power-Up: The voltage V_{DD} is switched on (Cold start). The CPU starts operation at page 15, PC = 00 when the internal timer oscillator started operation and has output 6 to 8 cycles.

- INITN Pin: Is held low for more than 10 µs (if this occurs during program execution it is named Warm start). The CPU starts operation at Page 15, PC = 00 when the INITN Pin is released to V_{DD} potential.

The TSS400(/4) internal registers have the following contents after the occurence of power up or the activation of the INIT Pin.

Register		Power up	INITN Pin
Program Counter	PC	00	00
Page Address	PA	15	15
Page Buffer	PB	15	15
Status	S	undefined	undefined
Special Status	SS	undefined	undefined
Accumulator	A	undefined	undefined
Y register	Y	undefined	undefined
X register	X	undefined	undefined
RAM Contents	(see Note)	undefined	unchanged
DAM Latch		undefined	undefined
ADD Latch		reset to 0	reset to 0
BRANCH Latch		reset to 0	reset to 0
Decimal Point	DP	reset to 0	reset to 0
Digit Latches	DLn	reset to 0	reset to 0
K-Port Latch		undefined	unchanged
I/O Port Data Latch		undefined	unchanged
I/O Port Control Latch		reset to 0	reset to 0
Timer0 and Timer1		undefined	unchanged
ADC Voltage SV_{DD}		switched off	switched off
LCD Segment Latches		undefined	unchanged
Subroutine Stack		level 0	level 0

Figure 11: Initialization Contents of internal Registers

Note: Despite the RAM remains unchanged if Warm start (see below) occurs, the memory addressed when the INITN Pin was activated may be destroyed by a write cycle.

Due to the undefined state of the status a BRANCH or CALL instruction located at page 15

PC = 00 (power up address) will not be executed surely. It is strongly recommended to use an instruction which is executed independently of the status at the initialization address (LDX, TCY e.g.).
The instruction located at page 15, PC = 00 maybe executed repeatedly. Thus toggle instructions (TDL, COMX8 e.g.) at this address should be ommitted.

If the TSS400(/4) system is battery powered and contains calibration factors or other important data in its RAM it is advised to distinguish between Cold start and Warm start. The reason is the possibility of initializations caused by EMI. If such an erroneous initialization is not tested for legality, EMI influence could destroy the RAM contents by clearing the RAM with the initialization software routine. Testing can be made by comparing RAM nibbles with known contents to their nominal value. These RAM nibbles could be identification codes or extra written test patterns (e.g. A5). If the tested RAM contains the right pattern, a spurious signal caused the initialization and normal program can continue. If the tested RAM nibbles differ from the nominal value, the RAM contents have been destroyed (e.g. by loss of power) and the initialization routine is invoked (clearing RAM, initializing computing constants). See 6.13 for software realization.

2.18 Hardware Options

The high grade of flexibility the TSS400(/4) offers makes it necessary to define which of the available options are used for a given application. With the hardware options it is possible to define completely the hardware for a TSS400(/4). A complete hardware definition is given in 2.18.15. See also "ADT User's Guide Compile Specification" for further details on putting together an actual specification list.

2.18.1 The Version Definition .OPTION VERSION

This option defines the product code for the TSS400(/4) mask. The code "yyy" is replaced by a 3 digit decimal number.

```
            .OPTION VERSION
            VERSION SYYY ON TSS40X
```

TSS40X describes which version of the TSS400 is used. The options are:

TSS400	TSS400/4
TSS402	TSS402/4
TSS403	TSS403/4
TSS404	TSS404/4
TSS405	TSS405/4

2.18.2 The Object Definition .OPTION ROM

This option precedes the object which is to be placed into the ROM of the TSS400(/4).

```
.OPTION ROM
TSS0400  OBJECT: EXAMPL CREATED 22 JUN 1987 10:29:21 VERS J.3 LB
0046000D002100400126014700E2000D0000008C008D00830000001270036019A


00BF005A006B009C01BC00060056008OO14C01FA004F01150094002301E800BF
```

2.18.3 The Pinning Definition .OPTION PINNING

This option precedes the definitions of the connections between the TSS400(/4) chip and the actual package pins. See 2.18.15 for an example of this option.

2.18.4 The K8 Flip Flop Definition .OPTION K8FF

This option defines if the K8 input goes to the ALU directly (FLIP-FLOP DISABLED) or if it is divided by a flip flop (FLIP-FLOP ACTIVE).

```
.OPTION K8FF                ; FLIP-FLOP IN K8-INPUT
FLIP-FLOP DISABLED          ; NORMAL K8 INPUT
FLIP-FLOP ACTIVE            ; K8 INPUT DIVIDED BY 2
```

2.18.5 The Pulldown Definition .OPTION K1K2K4K8IO

This option has two parts which are completely independent. The 1st part defines if a pulldown resistor is required for any input of the K-Port and the I/O pin. Allowed responses are YES and NONE. The 2nd part defines if the K-port, used as an output, has push-pull characteristics (NO) or open source characteristics (YES). Open source is recommended for keyboard scanning or other applications where a serial diode is needed normally. Push pull is recommended for driving of gates or switching of lines.

```
.OPTION K1K2K4K8IO
INPUT K1   PULL-DOWN  YES   ; K1 has pulldown resistance
INPUT K2   PULL-DOWN  NONE  ; K2 not
INPUT K4   PULL-DOWN  YES
INPUT K8   PULL-DOWN  YES
INPUT I/O  PULL-DOWN  NONE
OUPUT K1  OPEN-SOURCE NO    ; K1 has push-pull output stage
OUPUT K2  OPEN-SOURCE YES   ; K2 has open source output stage
OUPUT K4  OPEN-SOURCE NO
OUPUT K8  OPEN-SOURCE NO
```

2.18.6 The ADC Range Definition .OPTION ADC-Range

This option defines which range is to be used for the Analog Digital Converter. 4 different ranges are available, each with 4094 steps. Only one range is possible for all ADC inputs.

```
.OPTION ADC-RANGE
RANGE-IS LARGE              ; 0.1013 x SVDD - 0.4946 x SVDD
```

The other possible ADC ranges are:

```
RANGE-IS SMALL              ; 0.2313 x SVDD - 0.4075 x SVDD
RANGE-IS MEDIUM-LOW         ; 0.1003 x SVDD - 0.4025 x SVDD
RANGE-IS MEDIUM-HIGH        ; 0.2327 x SVDD - 0.5004 x SVDD
```

2.18.7 The Oscillator Speed Definition .OPTION OSCILLATOR

This option defines the source of the frequency for the MOS oscillator which is used for CPU timing. The item EXTERN means oscillator input via pin PFRQ which overwrites the internal oscillator.

```
.OPTION OSCILLATOR
FREQUENCY HIGH              ; 500 - 950 kHz INTERNAL
```

The other possible frequency ranges are:

```
FREQUENCY EXTERN            ; EXTERNALLY FED FREQUENCY
```

2.18.8 The Wake-up Frequency Definition .OPTION WAKE-UP-FREQUENCY.

This option defines the wakeup frequency which is active when DL12 is set (1). A timer oscillator frequenzy of 32768 Hz is assumed for the named wakeup frequencies.

```
.OPTION WAKE-UP-FREQUENCY
FREQUENCY 32HZ              ; Wakeup with 32 Hz
```

The other possible wakeup frequencies are:

```
FREQUENCY 8HZ               ; 8 Hz
FREQUENCY 16HZ              ; 16 Hz
FREQUENCY 64HZ              ; 64 Hz
FREQUENCY 128HZ             ; 128 Hz
```

2.18.9 The Output Frequency Definition .OPTION CLOCK

This option defines if a frequency which is derived from the timer oscillator is connected to pin CLOCK or if the pin is ommitted. Two frequencies are possible (a timer oscillator frequency of 32768 Hz is assumed):

```
.OPTION CLOCK
FREQUENCY 256HZ             ; 256 Hz connected to pin CLOCK
```

The other possible options are:

```
FREQUENCY 1024HZ            ; 1024 Hz connected to pin CLOCK
FREQUENCY NONE              ; Pin CLOCK is ommitted
```

2.18.10 The Timer Oscillator Definition .OPTION TIMOSC

This option defines the timer oscillator circuitry which is the time base for the display driver too. If no timer accuracy is needed the TRC option can be used which is cheap (only a resistor and a capacitor are needed) and has very fast start up features. If a display is used with this option, be sure that the worst case timer frequencies meet the limits of the display frame rate DFR:

$$DFR = \frac{\text{timer frequency}}{512}$$

If an accurate time base is necessary the option TXOSC and a quartz (10 to 100 kHz) are needed.

```
.OPTION TIMOSC
USE TRC                     ; RC DRIVEN TIMER OSCILLATOR
USE TXOSC                   ; QUARTZ DRIVEN OSCILLATOR
```

2.18.11 The LCD Multiplex Definition .OPTION COMMON

This option defines if 2 common multiplex or 4 common multiplex is used for the LCD display driver.

```
.OPTION COMMON
USE MUX4                    ; 4 common multiplex
```

The other possible option is:

```
USE MUX2                    ; 2 common multiplex
```

2.18.12 The Select Line Definition .OPTION SELECT-LINES

This option defines the controlling Y contents for each LCD digit and the common/select definition for each segment.

The 1st example shows the default definition for 4 common multiplex:

```
.OPTION SELECT-LINES                    ; 4 common multiplex
LOAD  S1+S2    BY  Y0                   ; Load digit by Y = 0
LOAD  S3+S4    BY  Y1
LOAD  S5+S6    BY  Y2
LOAD  S7+S8    BY  Y3
LOAD   S9+S10    BY  Y4
LOAD   S11+S12   BY  Y5
LOAD   S13+S14   BY  Y6
LOAD   S15+S16   BY  Y7
LOAD   S17+S18   BY  Y8
LOAD   S19+S20   BY  Y9                   ; Load digit by Y = 9
!                      COMMON  1 2 3 4

FOR  MUX4  AND  S2N-1  LOAD  A B C D      ; Odd select lines
FOR  MUX4  AND  S2N    LOAD  F G E H      ; Even select lines
```

The 2nd example shows the default definition for 2 common multiplex:

```
.OPTION SELECT-LINES                      ; 2 common multiplex
LOAD   S1+S2     BY  Y0                   ; Load digit by Y = 0
LOAD   S3+S4     BY  Y0
LOAD   S5+S6     BY  Y1
LOAD   S7+S8     BY  Y1
LOAD   S9+S10    BY  Y2
LOAD   S11+S12   BY  Y2
LOAD   S13+S14   BY  Y3
LOAD   S15+S16   BY  Y3
LOAD   S17+S18   BY  Y4
LOAD   S19+S20   BY  Y4
!                      COMMON  3 4

FOR  MUX2  AND  S4N-3  LOAD  B H          ; S1, S5, S9,  S13, S17
FOR  MUX2  AND  S4N-2  LOAD  A F          ; S2, S6, S10, S14, S18
FOR  MUX2  AND  S4N-1  LOAD  G E          ; S3, S7, S11, S15, S19
FOR  MUX2  AND  S4N    LOAD  C D          ; S4, S8, S12, S16, S20
```

2.18.13 The Output PLA Definition .OPTION SEGMENT-PLA

This option defines the output pattern for the LCD display. Which pattern is to be output depends on DL14, status and accumulator.

All 64 possibilities have to be defined. Only the segment H is controlled outside this option by the DP Latch. The segments are named as usual:

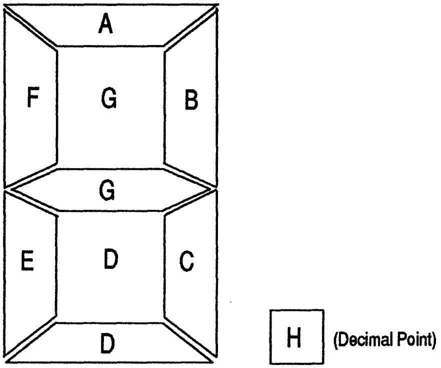


Figure 12: Standard Segment Notation Scheme

This is possible by defining the segments with the hardware option .OPTION SELECT LINES at the appropriate common/select combinations. See 2.18.12 for this definitions.

Example: A part of the 64 definitions is shown below.

```
CHARACTER 00 HAS-SEGMENTS NONE          ; DL14 = 0 , ST = 0
CHARACTER 01 HAS-SEGMENTS B C           ; 1
          .
CHARACTER 08 HAS-SEGMENTS A B C D E F G ; 8
          .
CHARACTER 15 HAS-SEGMENTS NONE          ; BLANK
          .
CHARACTER 63 HAS-SEGMENTS G             ; -
```

2.18.14 The .OPTION PACKAGE

The option PACKAGE defines which package is used. there are 3 packages available: 44PLCC, 40DIP, 28DIP

```
.OPTION PACKAGE
44PLCC
```

2.18.15 The End of Options Indication .END

This statement terminates the definition of the hardware options:

```
.END
```

2.18.16 Example for a complete TSS400(/4) Hardware Definition

```
!
.OPTION VERSION
VERSION S211 ON TSS400
!
.OPTION PINNING
!                        NO TEST      TEST   K        R       S/C
!                        KC = 0              RMIN     RMOUT   HVT
PAD  1  IS-PIN  1      APPL VSS      TEST VSS      VSS     VSS      ;
PAD  2  IS-PIN  2      APPL KC       TEST KC       KC      KC       ;
PAD  3  IS-PIN  3      APPL INITN    TEST INITN    INITN   INITN    ;
PAD  4  IS-PIN  4      APPL VDD      TEST VDD      VDD     VDD      ;
PAD  5  IS-PIN NONE APPL NONE        TEST NONE     R1     NONE PULLDO
PAD  6  IS-PIN  5      APPL SVDD     TEST SVDD     SVDD    SVDD     ;
PAD  7  IS-PIN  6      APPL RI       TEST RI       RI     RI    PULLDO
PAD  8  IS-PIN  7      APPL A4       TEST A4       A4     A4    PULLDO
PAD  9  IS-PIN NONE APPL NONE        TEST NONE     R2     NONE PULLDO
PAD 10  IS-PIN  8      APPL A3       TEST A3       A3     A3    PULLDO
PAD 11  IS-PIN NONE APPL NONE        TEST NONE     R3     NONE PULLDO
PAD 12  IS-PIN  9      APPL A2       TEST A2       A2     A2    PULLDO
PAD 13  IS-PIN NONE APPL NONE        TEST CLOCK    CLOCK CLOCK PULLD
PAD 14  IS-PIN 10      APPL A1       TEST A1       A1     A1    PULLDO
PAD 15  IS-PIN 11      APPL K8       TEST K8       NONE   NONE PULLDO
PAD 16  IS-PIN 12      APPL AGND     TEST AGND     AGND   AGND PULLDO
PAD 17  IS-PIN 13      APPL XOSCOUT  TEST XOSCOUT  XOSCOUT XOSCOUT ;
PAD 18  IS-PIN 14      APPL XOSCIN   TEST XOSCIN   XOSCIN  XOSCIN  ;
PAD 19  IS-PIN 15      APPL K4       TEST K4       NONE   NONE PULLDO
PAD 20  IS-PIN 16      APPL K1       TEST K1       NONE   NONE PULLDO
PAD 21  IS-PIN 17      APPL I/O      TEST I/O      I/O    I/O  PULLDO
PAD 22  IS-PIN 18      APPL R0       TEST NONE     R0      S20      ;
PAD 23  IS-PIN 19      APPL K2       TEST K2       NONE    S19      ;
PAD 24  IS-PIN 20      APPL R4       TEST NONE     R4      NONE     ;
PAD 25  IS-PIN NONE APPL R5          TEST NONE     R5      NONE     ;
PAD 26  IS-PIN 21      APPL S18      TEST NONE     R6      S18      ;
PAD 27  IS-PIN 22      APPL S17      TEST NONE     R7      S17      ;
PAD 28  IS-PIN 23      APPL S16      TEST NONE     NONE    S16      ;
PAD 29  IS-PIN 24      APPL S15      TEST NONE     NONE    S15      ;
PAD 30  IS-PIN 25      APPL S14      TEST NONE     NONE    S14      ;
PAD 31  IS-PIN 26   APPL S13      TEST S13     S13     S13     ;
PAD 32  IS-PIN 27   APPL S12      TEST S12     S12     S12     ;
PAD 33  IS-PIN 28   APPL S11      TEST S11     S11     S11     ;
PAD 34  IS-PIN 29   APPL S10      TEST COA4A1  MOSOSC  S10     ;
PAD 35  IS-PIN 30   APPL S9       TEST STA8A2  PC6     S9      ;
PAD 36  IS-PIN 31   APPL S8       TEST S8      S8      S8      ;
PAD 37  IS-PIN 32   APPL S7       TEST S7      S7      S7      ;
PAD 38  IS-PIN 33   APPL S6       TEST S6      S6      S6      ;
PAD 39  IS-PIN 34   APPL S5       TEST S5      S5      S5      ;
PAD 40  IS-PIN 35   APPL S4       TEST R036    R036    S4      ;
PAD 41  IS-PIN 36   APPL S3       TEST R147    R147    S3      ;
PAD 42  IS-PIN 37   APPL S2       TEST R258    R258    S2      ;
PAD 43  IS-PIN 38   APPL S1       TEST S1      S1      S1      ;
PAD 44  IS-PIN 39   APPL COM3     TEST COM3    COM3    COM3    ;
PAD 45  IS-PIN 40   APPL COM4     TEST COM4    COM4    COM4    ;
!
.OPTION PACKAGE
40DIP
!
.OPTION ROM
TSS0400  OBJECT:EXAMPL  CREATED 16 JUN 1987 10:29:21 VERS J.3 LB
0046000D002100400126014700E2000D0000008C008D0083000001270036019C
009800000100011E00450180000001E2019C0000000000450098009800880044
```

Note: Only four lines out of the 128 object lines are shown.

```
01E3009D008D017601150067000001FA001A0084000000360189014900480100
00BF005A006B009C01BC000600560080014C01FA004F011500940023 01E800BF
!
.OPTION K1K2K4K8IO
INPUT K1  PULL-DOWN  YES
INPUT K2  PULL-DOWN  NO
INPUT K4  PULL-DOWN  YES
INPUT K8  PULL-DOWN  YES
INPUT I/O PULL-DOWN  YES
OUTPUT K1 OPEN-SOURCE NO    ; PUSH-PULL
OUTPUT K2 OPEN-SOURCE NO    ; PUSH-PULL
OUTPUT K4 OPEN-SOURCE NO    ; PUSH-PULL
OUTPUT K8 OPEN-SOURCE NO    ; PUSH-PULL
!
.OPTION K8FF
FLIP-FLOP ACTIVE            ; FLIP FLOP IN K8-INPUT
!
.OPTION ADC-RANGE
RANGE-IS LARGE              ; 0.101 x SVDD - 0.49 x SVDD
!
```

```
.OPTION OSCILLATOR
FREQUENCY HIGH            ; INTERNAL OSCILATOR
!
.OPTION WAKE-UP-FREQUENCY
FREQUENCY 8HZ               ; 8 Hz wakeup for DL12 = 1
!
.OPTION CLOCK
FREQUENCY NONE
!
.OPTION TIMOSC
USE TXOSC                   ; QUARTZ IS USED
!
.OPTION COMMON
USE MUX2
.OPTION DELINIT IS ACTIVE
!
.OPTION SELECT-LINES         ; 2 COMMON STANDARD
LOAD  S1+S2    BY  Y0
LOAD  S3+S4    BY  Y0
LOAD  S5+S6    BY  Y1
LOAD  S7+S8    BY  Y1
LOAD  S9+S10   BY  Y2
LOAD  S11+S12  BY  Y2
LOAD  S13+S14  BY  Y3
LOAD  S15+S16  BY  Y3
LOAD  S17+S18  BY  Y4
LOAD  S19+S20  BY  Y4
!                    COMMON  1 2
FOR  MUX2  AND  S4N-3  LOAD  D H      ; Special select/common
FOR  MUX2  AND  S4N-2  LOAD  E C      ; combination
FOR  MUX2  AND  S4N-1  LOAD  G B
FOR  MUX2  AND  S4N    LOAD  F A
!
.OPTION SEGMENT-PLA
!        DL14 = 0 , ST = 0
CHARACTER 00 HAS-SEGMENTS NONE        ;
CHARACTER 01 HAS-SEGMENTS A ;
CHARACTER 02 HAS-SEGMENTS A B ;
CHARACTER 03 HAS-SEGMENTS A B C ;
CHARACTER 04 HAS-SEGMENTS A B C D ;
CHARACTER 05 HAS-SEGMENTS A B C D E ;
CHARACTER 06 HAS-SEGMENTS A B C D E F ;
CHARACTER 07 HAS-SEGMENTS A B C D E F G ;
CHARACTER 08 HAS-SEGMENTS B C D E F G ;
CHARACTER 09 HAS-SEGMENTS C D E F G ;
CHARACTER 10 HAS-SEGMENTS D E F G ;
CHARACTER 11 HAS-SEGMENTS E F G ;
CHARACTER 12 HAS-SEGMENTS F G ;
CHARACTER 13 HAS-SEGMENTS G ;
CHARACTER 14 HAS-SEGMENTS NONE
CHARACTER 15 HAS-SEGMENTS NONE
!
!        DL14 = 0 , ST = 1
CHARACTER 16 HAS-SEGMENTS A B C D E F ;         0
CHARACTER 17 HAS-SEGMENTS B C ;                 1
CHARACTER 18 HAS-SEGMENTS A B D E G ;           2
CHARACTER 19 HAS-SEGMENTS A B C D G ;           3
CHARACTER 20 HAS-SEGMENTS B C F G ;             4
CHARACTER 21 HAS-SEGMENTS A C D F G ;           5
CHARACTER 22 HAS-SEGMENTS A C D E F G ;         6
CHARACTER 23 HAS-SEGMENTS A B C ;               7
CHARACTER 24 HAS-SEGMENTS A B C D E F G ;       8
CHARACTER 25 HAS-SEGMENTS A B C D F G ;         9
CHARACTER 26 HAS-SEGMENTS A B C E F G ;         A
CHARACTER 27 HAS-SEGMENTS C D E F G ;           b
CHARACTER 28 HAS-SEGMENTS A D E F ;             C
CHARACTER 29 HAS-SEGMENTS B C D E G ;           d
CHARACTER 30 HAS-SEGMENTS A D E F G ;           E
CHARACTER 31 HAS-SEGMENTS A E F G ;             F
!
!        DL14 = 1 , ST = 0
CHARACTER 32 HAS-SEGMENTS B C E F G ;           H
CHARACTER 33 HAS-SEGMENTS B C D E ;             J
CHARACTER 34 HAS-SEGMENTS D E F ;               L
CHARACTER 35 HAS-SEGMENTS C E G ;               n
CHARACTER 36 HAS-SEGMENTS C D E G ;             o
CHARACTER 37 HAS-SEGMENTS A B E F G ;           P
CHARACTER 38 HAS-SEGMENTS E G ;                 r
CHARACTER 39 HAS-SEGMENTS D E F G ;             t
CHARACTER 40 HAS-SEGMENTS B C D E F ;           U
CHARACTER 41 HAS-SEGMENTS C D E ;               v
CHARACTER 42 HAS-SEGMENTS G ;                   -
CHARACTER 43 HAS-SEGMENTS D ;                   _
CHARACTER 44 HAS-SEGMENTS A B C D ;             ]
CHARACTER 45 HAS-SEGMENTS NONE
CHARACTER 46 HAS-SEGMENTS NONE
CHARACTER 47 HAS-SEGMENTS NONE
!
!        DL14 = 1 , ST = 1
CHARACTER 48 HAS-SEGMENTS A           ; SINGLE SEGMENTS
CHARACTER 49 HAS-SEGMENTS B
CHARACTER 50 HAS-SEGMENTS C
CHARACTER 51 HAS-SEGMENTS D
CHARACTER 52 HAS-SEGMENTS E
CHARACTER 53 HAS-SEGMENTS F
CHARACTER 54 HAS-SEGMENTS G
CHARACTER 55 HAS-SEGMENTS B C G         ; -1
CHARACTER 56 HAS-SEGMENTS B C D E F G ; INVERSE SINGLE SEGMENTS
CHARACTER 57 HAS-SEGMENTS A C D E F G
CHARACTER 58 HAS-SEGMENTS A B D E F G
CHARACTER 59 HAS-SEGMENTS A B C E F G
CHARACTER 60 HAS-SEGMENTS A B C D F G
CHARACTER 61 HAS-SEGMENTS A B C D E G
CHARACTER 62 HAS-SEGMENTS A B C D E F
CHARACTER 63 HAS-SEGMENTS NONE
.END
```

3 Standard Instruction Set Definitions

3.1 Effect on Status

Each instruction description in this section contains a status description. The way in which the instruction depends upon status or sets status is defined as follows:

- Set: The instruction is not conditional upon status and forces the status to one.
- Carry into status: The value of the carry from the adder is transferred to status. For the subtraction instructions, Carry = $\overline{\text{Borrow}}$.
- Comparison result: The logical comparison result from the ALU is transferred to status (equal: status is reset, unequal: status is set).
- Conditional on status: The instruction's execution results are conditional upon the state of the status. After executing the instruction, status is unconditionally set to one.
- Not applicable: Status not available due to nonoperating CPU.

3.2 Instruction Formats

3.2.1 Instruction Format I

This format has a two-bit opcode field, and the operand W is a seven bit ROM word address field. This format is used for program control by BRANCH and CALL instructions. The operand W, the branch address, has a value from 0 to >3F (127)

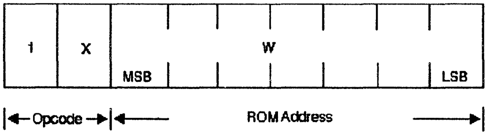


Figure 13: Instruction Format I

3.2.2 Instruction Format II

This format has a five-bit opcode field, and the operand C is a four-bit constant field. This format is used for instructions that contain an immediate value. The operand C has a value from 0 to 15.

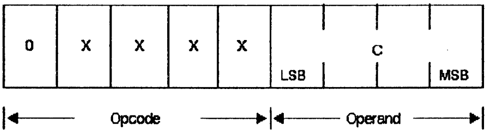


Figure 14: Instruction Format II

Note: The constant value C (from 0 to 15) is reversed in the C-field. The assembler properly converts any decimal value into this machine format. For example 3 (0011) is converted to 12 (1100).

3.2.3 Instruction Format III

This format has a seven-bit opcode field, and the operand B is a two-bit constant field. This format is used for instructions that address bits in a nibble. The operand B has a value from 0 to 3.

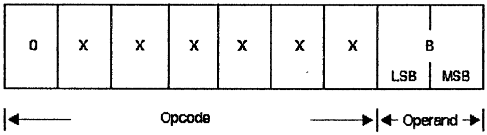


Figure 15: Instruction Format III

Note: The constant value B (from 0 to 3) is reversed in the B-field. The assembler properly converts any decimal value into this machine format. For example 2 (10) is converted to 1 (01).

3.2.4 Instruction Format IV

This format defines a nine-bit operation code field only. Instructions of this format have no constant operands. The instruction always performs the same action, for example transferring the accumulator to the Y register.

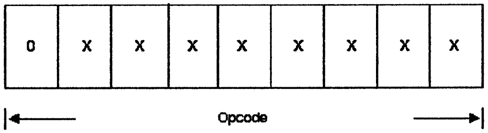


Figure 16: Instruction Format IV

3.2.5 Coding Format

The following rules should be followed in writing a program:

- Label fields are a maximum of eight alphanumeric characters starting with an alphabetic character. The label field begins in column one.
- The operation code is to the right of a label, the two separated by at least one blank space. If no label is used, the operation code begins after the first column (second column or further over).
- The operand is to the right of the operation code, the two separated by at least one blank space.
- A comment is to the right of the operand, the two separated by at least one blank space. If a comment occupies a separate line, it must begin with an asterisk (*) in column one.

For legibility, it is recommended that the fields begin in the following columns:

- label fields begin in column 1
- operation codes should begin in columnn 11
- operands should begin in column 21
- comments to an instruction should begin in column 31
- comment lines begin in column one with an asterisk

Example: The following lines show source lines conforming to the above rules:

```
* THIS IS A PURE COMMENT LINE
LABEL         OPCODE        OPERAND       COMMENT
|             |             |             |
|             |             |             column 31
|             |             column 21
|             column 11
column 1
```

3.2.6 Micro Instructions

The TSS400(/4) uses microcoded instructions and fixed instructions. If microinstructions are used, they are listed after the description of the instruction. This can be helpful in understanding the effects of an instruction.

The used microinstructions are:

STO	Accumulator moved to addressed memory
CKM	CKB logic output to addressed memory
@CKP	CKB logic connected to P adder input
@YTP	Y register connected to P adder input
@MTP	Addressed memory connected to P adder input
@ATN	Accumulator connected to N adder input
@NATN	Inverted accumulator connected to N adder input
@MTN	Addressed memory connected to N adder input
@15TN	Constant 15 (-1) connected to N adder input
@CKN	CKB logic connected to N adder input
NE	Comparator output written to status
C8	Carry out of MSB written to status
@CIN	Constant 1 added
AUTA	ALU output written to accumulator
AUTY	ALU output written to Y register
@NDMTP	Inverted DAM connected to P adder input
@DMTP	DAM connected to P adder input
SSE	Special Status set or reset according to result

SSS	Special Status added to adder inputs
CME	Accumulator moved to memory conditionally

The CKM, @CKP and @CKN microinstructions depend on the opcode. The actual opcode determines which data the CKB bus will have on it when these microinstructions are called. See Figure 17 for more information.

Opcode	Source Data on CKB bus
008 - 00F	K-lines
018 - 01F	Timer
020 - 02F	Bit data TBIT, TCTM, TRTM
0A0 - 0AF	Bit data SBIT, RBIT, TBITA, TIOM
040 - 07F	Constant data from instruction
0C0 - 0FF	Constant data from instruction

Figure 17: CKB Memory Map Description

3.3 Register to Register Transfer Instructions

3.3.1 Transfer Accumulator to Y Register

MNEMONIC: TAY OPCODE: 038

STATUS: set FORMAT: IV

ACTION: A → Y

PURPOSE: Copying or saving the accumulator.

DESCRIPTION: The accumulator contents are unconditionally transferred to the Y register. The accumulator contents are unaltered.

MICROINSTRUCTIONS: @ATN, AUTY

3.3.2 Transfer Y Register to Accumulator

MNEMONIC: TYA OPCODE: 03B

STATUS: set FORMAT: IV

ACTION: Y → A

PURPOSE: Copying or saving the Y register.

DESCRIPTION: The four-bit contents of the Y register are unconditionally transferred to the accumulator. The contents of the Y register are unaltered.

MICROINSTRUCTIONS: @YTP, AUTA

3.3.3 Clear Accumulator

MNEMONIC: CLA OPCODE: 006

STATUS: set FORMAT: IV

ACTION: 0 → A

DESCRIPTION: The contents of the accumulator are unconditionally cleared to zero.

MICROINSTRUCTIONS: AUTA

3.4 Register to Memory, Memory to Register Transfer Instructions

3.4.1 Transfer DAM to Accumulator

MNEMONIC: TDA OPCODE: 016

STATUS: set FORMAT: IV

ACTION: M(15,Y) → A

PURPOSE: To transfer the DAM value to accumulator without changing the X register.

DESCRIPTION: The contents of the DAM location addressed by the Y register are loaded into the accumulator. The X register and the memory contents are not altered.

MICROINSTRUCTIONS: @DMTP, AUTA

EXAMPLE: Comparison of DAM to memory

```
LDX       1
TCY       3          ADDRESS M(1,3)
TDA                  M(15,3) -> A
MNEA                 M(15,3) = M(1,3) ?
BRANCH    NOTEQU     NO, NOT EQUAL
...                  EQUAL
```

3.4.2 Exchange DAM and Accumulator

MNEMONIC: XDA OPCODE: 019

STATUS: set FORMAT: IV

ACTION: M(15,Y) ↔ A

PURPOSE: To exchange DAM value and accumulator without changing the X register.

DESCRIPTION: The contents of the DAM location addressed by the Y register and the contents of the accumulator are exchanged. The X register is not altered.

MICROINSTRUCTIONS: @DMTP, STO, AUTA

EXAMPLE: Exchange M(1,0-5) and M(15,0-5)

```
          LDX       1          ADDRESS M(1,5)
          TCY       5
L$300     TMA                  M(1,Y) -> A
          XDA                  A <-> M(15,Y)
          TAMDYN               A -> M(1,Y)
          BRANCH    L$300      LOOP UNTIL Y = 15
```

3.4.3 Transfer Accumulator to Memory

MNEMONIC: TAM OPCODE: 03F

STATUS: set FORMAT: IV

ACTION: A → M(X,Y)

PURPOSE: Storing the accumulator in the RAM.

DESCRIPTION: The four-bit contents of the accumulator are stored in the memory location addressed by the X and Y registers. The accumulator contents are unaltered.

MICROINSTRUCTIONS: STO

3.4.4 Transfer Accumulator to Memory and Increment Y Register

MNEMONIC: TAMIYC OPCODE: 03D

STATUS: carry into status FORMAT: IV

ACTION: A → M(X,Y)
Y + 1 → Y
1 → S if Y = 0
0 → S if Y # 0

PURPOSE: The Y register sequentially addresses a bank of 16 RAM words, and the addressed words are set to the accumulator value(s), during initialization for example.

DESCRIPTION: The contents of the accumulator are stored in the memory location addressed by the X and Y registers. The contents of the accumulator are unaltered. Then the contents of the Y register are incremented by one. Carry information is transferred into status. If the result is zero, status is set.

MICROINSTRUCTIONS: STO, @YTP, @CIN, C8, AUTY

3.4.5 Transfer Accumulator to Memory and Decrement Y Register

MNEMONIC: TAMDYN OPCODE: 03C

STATUS: carry into status FORMAT: IV

ACTION:
$A \rightarrow M(X,Y)$
$Y - 1 \rightarrow Y$
$1 \rightarrow S$ if $Y \# 15$
$0 \rightarrow S$ if $Y = 15$

PURPOSE: The Y register sequentially addresses a bank of 16 RAM words, and the addressed words are set to the accumulator value(s), during initialization for example.

DESCRIPTION: The contents of the accumulator are stored in the memory location addressed by the X and Y registers. The contents of the accumulator are unaltered. Then the contents of the Y register are decremented by one. Carry information is transferred to status. If the result is not equal to 15, status will be set.

MICROINSTRUCTIONS: STO, @YTP, @15TN, C8, AUTY

3.4.6 Tranfer Accumulator to Memory and Zero Accumulator

MNEMONIC: TAMZA OPCODE: 03E

STATUS: set FORMAT: IV

ACTION:
$A \rightarrow M(X,Y)$
$0 \rightarrow A$

PURPOSE: Storing and clearing of accumulator in one instruction.

DESCRIPTION: The contents of the accumulator are stored in the RAM location addressed by the X and Y registers. The contents of the accumulator are then cleared to zero.

MICROINSTRUCTIONS: STO, AUTA

3.4.7 Transfer Memory to Y Register

MNEMONIC: TMY OPCODE: 03A

STATUS: set FORMAT: IV

ACTION: M(X,Y) → Y

PURPOSE: Loading of Y register from memory. This is useful if the Y register is used as a pointer.

DESCRIPTION: The contents of the RAM location addressed by the X and Y registers are loaded into the Y register. Memory contents are unaltered.

MICROINSTRUCTIONS: @MTP, AUTY

3.4.8 Transfer Memory to Accumulator

MNEMONIC: TMA OPCODE: 039

STATUS: set FORMAT: IV

ACTION: M(X,Y) → A

PURPOSE: Loading accumulator from memory.

DESCRIPTION: The contents of the RAM location addressed by X and Y registers are loaded into the accumulator. Memory contents are unaltered.

MICROINSTRUCTIONS: @MTP, AUTA

3.4.9 Exchange Memory and Accumulator

MNEMONIC: XMA OPCODE: 003

STATUS: set FORMAT: IV

ACTION: M(X,Y) ↔ A

PURPOSE: Exchange of accumulator and memory without intermediate storage.

DESCRIPTION: The memory contents (addressed by the X and Y registers) are exchanged with the accumulator contents. For example, this instruction is useful to retrieve a memory word into the accumulator for an arithmetic operation and save the current accumulator in this RAM location temporarily. The accumulator can be restored by a second XMA instruction.

MICROINSTRUCTIONS: @MTP, STO, AUTA

3.5 Arithmetic Instructions

3.5.1 Add Constant to Accumulator, Result to Accumulator

MNEMONIC:	ACACC C	OPCODE: 070 - 07F
STATUS:	carry into status	FORMAT: II

ACTION: A + C → A
1 → S if sum > 15
0 → S if sum < 16

PURPOSE: To add a constant to the accumulator.

DESCRIPTION: The constant from the four lower bits of the instruction, is added to the accumulator contents. Carry information is transferred into status. A sum greater than 15 will generate a carry and will set status.

MICROINSTRUCTIONS: @CKP, @ATN, C8, AUTA

3.5.2 Add Memory to Accumulator, Result to Accumulator

MNEMONIC:	AMAAC	OPCODE: 015
STATUS:	carry into status	FORMAT: IV

ACTION: $M(X,Y) + A \rightarrow A$
$1 \rightarrow S$ if sum > 15
$0 \rightarrow S$ if sum < 16

PURPOSE: Addition of memory contents to accumulator.

DESCRIPTION: The contents of the memory location addressed by the X and Y registers are added to the contents of the accumulator. The result is stored in the accumulator. The resulting carry information is transferred to status. A sum that is greater than 15 results in a carry and a one to status. Memory contents are unaltered.

MICROINSTRUCTIONS: @MTP, @ATN, C8, AUTA

3.5.3 Substract Accumulator from Memory, Result to Accumulator

MNEMONIC: SAMAN OPCODE: 030

STATUS: carry into status FORMAT: IV

ACTION: $M(X,Y) - A \rightarrow A$
$1 \rightarrow S$ if $A \leq M(X,Y)$ Initial conditions
$0 \rightarrow S$ if $A > M(X,Y)$

PURPOSE: Subtracting accumulator from memory.

DESCRIPTION: The contents of the accumulator are subtracted from the memory word addressed by the X and Y registers via two's complement addition. The result is stored into the accumulator. Status is set if the accumulator is less than or equal to the memory word, indicating that no-borrow occurred. A borrow occurs when the accumulator is greater than the memory word and status is reset to zero.

MICROINSTRUCTIONS: @MTP, @NATN, @CIN, C8, AUTA

3.5.4 Increment Memory and load into Accumulator

MNEMONIC: IMAC OPCODE: 032

STATUS: carry into status FORMAT: IV

ACTION: M(X,Y) + 1 → A
1 → S if M(X,Y) = 15
0 → S if M(X,Y) < 15

PURPOSE: Loading and incrementing accumulator in one instruction. Useful for incrementing counters.

DESCRIPTION: The contents of memory addressed by the X and Y registers are fetched. One is added to this word and the result is stored in the accumulator. The resulting carry information is transferred to status. Status is set if the sum is greater than 15. Memory is left unaltered.

MICROINSTRUCTIONS: @MTP, @CIN, C8, AUTA

Example: The subroutine below increments a counter M(X,0-3) in hexadecimal format.

```
INCR     LDX        X          ADDRESS LSD M(X,3)
         TCY        3
L$102    IMAC                  INCR. DIGIT
         BRANCH     L$101      IF CARRY, INCR. NEXT DIGIT
         TAM                   NO CARRY, STORE RESULT
         RETN                  RETURN TO MAIN PROGRAM
L$101    TAMDYN                CARRY: STORE RESULT. Y = Y-1
         BRANCH     L$102      INCR. NEXT DIGIT IF MSD NOT
         RETN                  YET REACHED.
```

3.5.5 Decrement Memory and load into Accumulator

MNEMONIC: DMAN OPCODE: 007

STATUS: carry into status FORMAT: IV

ACTION: M(X,Y) - 1 -> A
1 → S if M(X,Y) # 0
0 → S if M(X,Y) = 0

PURPOSE: Loading and decrementing accumulator in one instruction. Useful for decrementing counters.

DESCRIPTION: The contents of memory addressed by the X and Y registers are fetched. One is subtracted from this word (add 15), and the result is placed in the accumulator. The resulting carry information is transferred to status. If memory is greater than or equal to one, status is set. Memory contents are unaltered.

MICROINSTRUCTIONS: @MTP, @15TN, C8, AUTA

3.5.6 Increment Y Register

MNEMONIC: IYC OPCODE: 005

STATUS: carry into status FORMAT: IV

ACTION: Y + 1 → Y
1 → S if Y = 0
0 → S if Y # 0

PURPOSE: Incrementing Y register e.g. for loop control.

DESCRIPTION: The contents of the Y register are incremented by one. The result is placed back into the Y register. Resulting carry information is transferred to status. A sum greater than 15 results in status being set.

MICROINSTRUCTIONS: @YTP, @CIN, C8, AUTY

3.5.7 Decrement Y Register

MNEMONIC: DYN OPCODE: 004

STATUS: carry into status FORMAT: IV

ACTION: Y - 1 → Y
1 → S if Y # 15
0 → S if Y = 15

PURPOSE: To decrement the contents of the Y register by one.

DESCRIPTION: The contents of the Y register are decremented by one. This is performed by adding a minus one (15). Resulting carry information is transferred into status. If the result is not equal to 15, status will be set.

MICROINSTRUCTIONS: @YTP, @15TN, C8, AUTY

3.5.8 Add Constant to Y Register

MNEMONIC: ACYY C OPCODE: 0F0 - 0FF

STATUS: set FORMAT: II

ACTION: Y + C → Y

PURPOSE: Adding a constant to Y register. Useful in transfer loops with different Y addresses for digits.

DESCRIPTION: The four-bit constant of the C field is added to the contents of the Y register. Status is always set.

MICROINSTRUCTIONS: @YTP, @CKN, AUTY

Example: The subroutine transfers 4 nibbles from M(1,0-3) to M(2,4-7).

```
RAMXFER   TCY       0         ADDRESS SOURCE MSD
L$104     LDX       1
          TMA                 SOURCE DIGIT -> A
          LDX       2
          ACYY      4         ADD Y DIFFERENCE TO Y
          TAM                 DIGIT -> DESTINATION
          ACYY      13        ADDRESS NEXT SOURCE DIGIT
          YNEC      4         SOURCE TRANSFERRED ?
          BRANCH    L$104     NO, PROCEED
          RETN
```

3.5.9 Complement Accumulator and Increment

MNEMONIC: CPAIZ OPCODE: 031

STATUS: carry into status FORMAT: IV

ACTION: $\bar{A} + 1 \rightarrow A$
$1 \rightarrow S$ if $A = 0$
$0 \rightarrow S$ if $A \# 0$

PURPOSE: To obtain the two's complement of the accumulator.

DESCRIPTION: The two's complement of the accumulator is computed by adding one to the one's complement of the accumulator and storing the result in the accumulator. Carry information is transferred into status. If the accumulator contents are zero, carry occurs, and status is set.

MICROINSTRUCTIONS: @NATN, @CIN, C8, AUTA

3.6 Special Arithmetic Instructions

The Special Arithmetic Instructions result from the need for faster computing, especially multiplying. The instructions are tailored to fast addition and subtraction and are not easy to understand. The purpose is minimizing the instruction count inside the add or subtract loops of multiplication and division.

3.6.1 Add DAM and Memory and Special Status, Result to Accumulator

MNEMONIC: DMEA OPCODE: 010

STATUS: set FORMAT: IV

ACTION: M(15,Y) + M(X,Y) + SS → AC

PURPOSE: Fast addition without X register changes.

DESCRIPTION: The contents of the DAM and of the addressed memory are added together with the Special Status generated before. The result is stored in the accumulator. The carry is stored in the ALU.

Note: If DAM is addressed, the doubled contents of M(15,Y) are added to the special status.

MICROINSTRUCTIONS: @MTN, @DMTP, SSS, AUTA

Example: Addition of M(15,0-8) to M(3,0-8)

```
ADDITION   SAL                  PREPARE ADD LATCH FOR ADDITION
           REAC                 RESET SS
           LDX        3         M(3,8-0) + M(15,8-0)
           TCY        8         LSD M(X,8)
ADDLOOP    DMEA                 M(3,Y) + M(15,Y) + SS
           TAMACS     6         TEST IF CORRECTION NECESSARY
           CTMDYN               CORRECT RESULT. Y = Y-1
           BRANCH     ADDLOOP   LOOP UNTIL MSD IS HANDLED TOO
           RETN                 RESULT IN M(3,8-0)
```

3.6.2 Add inverted DAM and Memory and Special Status, Result to Accumulator

MNEMONIC: NDMEA OPCODE: 013

STATUS: set FORMAT: IV

ACTION: $\overline{M(15,Y)} + M(X,Y) + SS \rightarrow AC$

PURPOSE: Fast subtraction without changes of the X register.

DESCRIPTION: The contents of the inverted DAM and of the addressed memory are added together with the Special Status generated before. The result is stored in the accumulator. The carry is stored in the ALU.

MICROINSTRUCTIONS: @MTN, @NDMTP, SSS, AUTA

Example: Subtract M(15,0-8) from M(3,0-8)

```
SUBTRACT   SEAC                 SET SS. ADD LATCH IS RESET
           LDX        3         M(3,8-0) - M(15,8-0)
           TCY        8                    ________
SUBLOOP    NDMEA                M(3,Y) + M(15,Y) + SS
           TAMACS     10        TEST IF CORRECTION NECESSARY
           CTMDYN               CORRECT RESULT. Y = Y-1
           BRANCH     SUBLOOP   LOOP UNTIL MSD IS DONE
           RETN                 RESULT IN M(3,8-0)
```

3.6.3 Add DAM and inverted Accumulator and Special Status, Result to Accumulator

MNEMONIC: DNAA OPCODE: 011

STATUS: set FORMAT: IV

ACTION: $M(15,Y) + \overline{A} + SS \rightarrow A$

PURPOSE: To subtract the accumulator from a DAM nibble.

DESCRIPTION: The addressed DAM and the inverted accumulator are added together with the Special Status. The result is stored in the accumulator. The carry is stored in the ALU.

MICROINSTRUCTIONS: @DMTP, @NATN, SSS, AUTA

Example: The routine subtracts the accumulator from M(15,0-8) in BCD format.

```
SUBACC    SEAC                SET SS FOR SUBTRACTION
          TCY       8         ADDRESS LSD, X CAN BE RANDOM
L$360     DNAA                SUBTRAHEND IN A: DAM - A
          TAMACS    10        RESULT -> M(15,Y)
          CTMDYN              CORRECT IF NECESSARY
          CCLA                SS -> A
          YNEC      15        ALL THROUGH ?
          BRANCH    L$360     LOOP UNTIL MSD IS HANDLED TOO
          RETN
```

3.6.4 Reset End-Around-Carry [Special Status]

MNEMONIC: REAC OPCODE: 0B4

STATUS: set FORMAT: IV

ACTION: $0 \rightarrow SS$

PURPOSE: To reset the Special Status before entering and add loop.

DESCRIPTION: The Special Status is reset.

Note: The setting of the ADD Latch by the instruction "SAL" can cause ambiguous change of the Special Status. So it is advised to set or reset the Special Status after the SAL instruction. See example at 3.6.1

MICROINSTRUCTIONS: none

3.6.5 Set End-Around-Carry [Special Status]

MNEMONIC: SEAC OPCODE: 0B5

STATUS: set FORMAT: IV

ACTION: 1 → SS

PURPOSE: To set the Special Status before entering a subtract loop.

DESCRIPTION: The Special Status is set.

Note: The setting of the ADD Latch by the instruction "SAL" can cause ambiguous change of the Special Status. So it is advised to set or reset the Special Status after the SAL instruction. See example at 3.6.1

MICROINSTRUCTIONS: none

3.6.6 Set ADD Latch

MNEMONIC: SAL OPCODE: 0B1

STATUS: set FORMAT: IV

ACTION: 1 → ADD Latch

PURPOSE: Preparing of the Special Status logic for addition. (If not set, the adder logic is prepared for subtraction). See examples of Special Arithmetic Instructions.

DESCRIPTION: The "ADD Latch" is set (until a RETN instruction is performed).

MICROINSTRUCTIONS: none

3.6.7 Transfer Accumulator to Memory, add Constant to Accumulator

MNEMONIC: TAMACS C OPCODE: 0D0 - 0DF

STATUS: set FORMAT: II

ACTION:
A → M(X,Y)
A + C → A
If Add Latch = 1:
If Prev. Carry .OR. Carry = 1: 1 → SS
Otherwise 0 → SS
If Add Latch = 0:
Prev. Carry → SS

PURPOSE: Transfer of result and correction test for addition and subtraction in one instruction. See examples of Special Arithmetic Instructions.

DESCRIPTION: The four-bit contents of the accumulator are stored in the memory location adressed by the X and Y registers. Then the four bit constant from the four lower bits of the instruction is added to the accumulator . If the ADD Latch is set and the carry from the previous or the TAMACS instruction is 1, then the Special Status is set. If the ADD Latch is reset and the carry from the previous instruction is 1, then the Special Status is set. If the conditions above are not satisfied, the Special Status is reset.

MICROINSTRUCTIONS: STO, @ATN, @CKP, AUTA, SSE

3.6.8 Conditionally Transfer Accumulator to Memory, Decrement Y Register

MNEMONIC: CTMDYN OPCODE: 018

STATUS: carry into status FORMAT: IV

ACTION: If ADD Latch = 1 .AND. SS = 1: A → M(X,Y)
If ADD Latch = 0 .AND. SS = 0: A → M(X,Y)
Y - 1 → Y
If Y # 15: 1 → S
If Y = 15: 0 → S

PURPOSE: Transfer of corrected result when correction (e.g. for decimal) was necessary. Decrement of Y register for looping until MSD M(X,0) is handled too. See examples for Special Arithmetic Instructions.

DESCRIPTION: If both ADD Latch and Special Status are set or both are reset then the four-bit contents of the accumulator are stored in the memory location adressed by the X and Y registers. Then the contents of the Y register are decremented by one. This is performed by adding a minus one (15). Resulting carry information is transferred into status. If the result is not equal to 15, status will be set.

MICROINSTRUCTIONS: @YTP, @15TN, C8, AUTY, CME

3.6.9 Transfer Special Status to Accumulator

MNEMONIC: CCLA OPCODE: 012

STATUS: set FORMAT: IV

ACTION: SS → A

PURPOSE: Makes the Special Status visible e.g. for overflow tests out of the MSD.

DESCRIPTION: The Special Status is transferred into the least significant bit (LSB) of the accumulator. The other accumulator bits are reset.

MICROINSTRUCTIONS: AUTA, SSS

3.6.10 Set Branch Latch

MNEMONIC: SBL OPCODE: 0B3

STATUS: set FORMAT: IV

ACTION: 1 → BRANCH Latch

PURPOSE: For very fast additions or subtractions by decoding of branches as instructions and branches.

DESCRIPTION: The BRANCH Latch is set (until a RETN instruction is performed).

MICROINSTRUCTIONS: none

3.7 Arithmetic Compare Instructions

3.7.1 If Accumulator is less than or equal to Memory, One to Status

MNEMONIC: ALEM OPCODE: 001

STATUS: carry into status FORMAT: IV

ACTION: 1 → S if $A \leq M(X,Y)$
0 → S if $A > M(X,Y)$

PURPOSE: Comparing accumulator and addressed memory.

DESCRIPTION: The value from the accumulator is subtracted from the contents of the memory location, addressed by the X and Y registers, using two's complement addition. Resulting carry information is transferred into status. Status equal to one indicates that the accumulator is less than or equal to the memory word. Memory and accumulator contents are unaltered.

MICROINSTRUCTIONS: @MTP, @NATN, @CIN, C8

3.7.2 If Accumulator is less than or equal to Constant, One to Status

MNEMONIC: ALEC C OPCODE: 0E0 - 0EF

STATUS: carry into status FORMAT: II

ACTION: $1 \rightarrow S$ if $A \leq C$
$0 \rightarrow S$ if $A > C$

PURPOSE: To arithmetically compare accumulator contents to a constant value.

DESCRIPTION: The accumulator value is subtracted from the constant (in C field of the instruction) using two's complement addition. Resulting carry information is transferred into status. Status is set if the accumulator is less than or equal to the constant. The accumulator data is unaltered.

MICROINSTRUCTIONS: @CKP, @NATN, @CIN, C8

3.8 Logical Compare Instructions

3.8.1 If Memory is not equal to Accumulator, One to Status

MNEMONIC: MNEA OPCODE: 009

STATUS: comparison result FORMAT: IV

ACTION: $1 \rightarrow S$ if $M(X,Y) \# A$
$0 \rightarrow S$ if $M(X,Y) = A$

PURPOSE: To compare the addressed memory with the accumulator contents.

DESCRIPTION: The contents of the memory addressed by the X and Y registers are logically compared to the accumulator contents. The comparison information is transferred to status. Inequality will set status.

MICROINSTRUCTIONS: @MTP, @ATN, NE

3.8.2 If Memory is not equal to Zero, One to Status

MNEMONIC: MNEZ OPCODE: 033

STATUS: comparison result FORMAT: IV

ACTION: 1 → S if M(X,Y) # 0
0 → S if M(X,Y) = 0

PURPOSE: To compare a memory word to zero.

DESCRIPTION: The memory contents addressed by the X and Y register are logically compared to zero. Comparison information is transferred into status. Inequality between memory value and zero will set status.

MICROINSTRUCTIONS: @MTP, NE

3.8.3 If Y Register is not equal to Accumulator, One to Status

MNEMONIC: YNEA OPCODE: 002

STATUS: comparison result FORMAT: IV

ACTION: 1 → S if Y # A
0 → S if Y = A

PURPOSE: To compare the contents of the Y register and the accumulator for inequality.

DESCRIPTION: The contents of the Y register are logically compared to the contents of the accumulator. Comparison information is transferred into status. Inequality will set status.

MICROINSTRUCTIONS: @YTP, @ATN, NE

3.8.4 If Y Register is not equal to a Constant, One to Status

MNEMONIC: YNEC C OPCODE: 050 - 05F

STATUS: comparison result FORMAT: II

ACTION: 1 → S if Y # C
0 → S if Y = C

PURPOSE: Comparing Y register to a constant value. Useful for loop control.

DESCRIPTION: The contents of the Y register are logically compared to the four-bit value from the C field of the instruction. Compare result is transferred into status. Inequality between the operands causes status to be set.

MICROINSTRUCTIONS: @YTP, @CKN, NE

Example: The subroutine clears the memory M(X,0-4) in a loop.

```
CLEAR    TCY      0       START WITH M(X,0)
L$103    TCMIY    0       CLEAR NIBBLE AND INCR. Y
         YNEC     5       Y > YMAX ?
         BRANCH   L$103   IF NOT CLEAR NEXT NIBBLE
         RETN             Y = 5: RETURN TO CALLER
```

3.9 Bit Manipulation Instructions

3.9.1 Set Memory Bit

MNEMONIC: SBIT B OPCODE: 0A0 - 0A3

STATUS: set FORMAT: III

ACTION: 1 → M(X,Y,B)

PURPOSE: Setting a single bit without affecting the other bits in the addressed nibble. Useful for flags.

DESCRIPTION: One of the four bits, as selected by the B-field of the operand, is set to one in the memory word addresssed by the contents of the X and Y registers.

MICROINSTRUCTIONS: none

3.9.2 Reset Memory Bit

MNEMONIC: RBIT B OPCODE: 0A4 - 0A7

STATUS: set FORMAT: III

ACTION: 0 → M(X,Y,B)

PURPOSE: Resetting a single bit without affecting the other bits in the addressed nibble. Useful for flags.

DESCRIPTION: One of the four bits, as selected by the B field of the instruction, is reset to zero in the memory word addressed by the contents of the X and Y registers.

MICROINSTRUCTIONS: none

3.9.3 Test Memory Bit

MNEMONIC: TBIT B OPCODE: 020 - 023

STATUS: comparison result FORMAT: III

ACTION: M(X,Y,B) → S

PURPOSE: To test a selected memory bit for a logic one and set status accordingly e.g. testing of flags.

DESCRIPTION: The addressed memory bit is transferred to the status. This means, if the memory bit is set, the status is set to one, if the memory bit is not set, the status is reset to zero.

MICROINSTRUCTIONS: @CKP, @CKN, @MTP, NE

3.9.4 Test Accumulator Bit for One

MNEMONIC: TBITA B OPCODE: 0A8 - 0AB

STATUS: comparison result FORMAT: III

ACTION: If accumulator bit = 1: 1 → S
If accumulator bit = 0: 0 → S

PURPOSE: To test a selected accumulator bit for a logic one.

DESCRIPTION: The value of the accumulator bit defined by the operand is transferred to status.

MICROINSTRUCTIONS: @ATN, @CKN, @CKP, NE

3.9.5 Or Memory with Accumulator, Result to Accumulator

MNEMONIC: ORMA OPCODE: 014

STATUS: set FORMAT: IV

ACTION: M(X,Y) .OR. A → A

PURPOSE: Logical OR function

DESCRIPTION: The addressed memory is ORed bitwise with the accumulator. The result is stored in the accumulator.

MICROINSTRUCTIONS: @ATN, @MTN, AUTA

Example: The logical OR of M(4,2) and the accumulator is built and tested if bit 2 is set in one of them.

```
LDX       4
TCY       2
ORMA                A .OR. M(X,Y)
TBITA     2         IS BIT 2 SET IN one OF THEM ?
BRANCH    L$222     YES, BIT 2 OF A IS 1
...                 NO
```

Example: Test if bit 1 and bit 2 of M(X,Y) are set together. A mask is used with set bits at the positions which are not interesting. When all four accumulator bits are set after the ORMA instruction both bits were set in M(X,Y).

```
TCA       9         BUILD MASK
ORMA                A .OR. M(X,Y)
ACACC     1         M(X,Y,1) .AND. M(X,Y,2) = 1 ?
BRANCH    L$223     YES, A = 15
...                 NO
```

3.9.6 Or Memory with Inverted Accumulator, Result to Accumulator

MNEMONIC: ORMNAA OPCODE: 017

STATUS: set FORMAT: IV

ACTION: M(X,Y) .OR. $\bar{A} \rightarrow A$

PURPOSE: Logical NOR function (masks e.g.).

DESCRIPTION: The addressed memory is ORed bitwise with the inverted accumulator. The result is stored in the accumulator.

MICROINSTRUCTIONS: @NATN, @MTN, AUTA

Example: Test if bit 1 and bit 2 of M(X,Y) are set together. A mask is used in the accumulator which defines the interesting bits.

```
         TCA        6         BUILD MASK BIT 1 .OR. BIT 2
*                                            _
         ORMNAA               M(X,Y) .OR. A
         ACACC      1         M(X,Y,1) .AND. M(X,Y,2) = 1 ?
         BRANCH     L$224     YES, IF A = 15
         ...                  NO
```

3.10 Constant Transfer Instructions

3.10.1 Transfer Constant to Accumulator

MNEMONIC: TCA C OPCODE: 0C0 - 0CF

STATUS: set FORMAT: II

ACTION: C → A

PURPOSE: Loading accumulator with a constant in one instruction.

DESCRIPTION: The four-bit constant from the C-field of the instruction is loaded into the accumulator.

MICROINSTRUCTIONS: @CKP, AUTA

3.10.2 Transfer Constant to Y Register

MNEMONIC: TCY C   OPCODE: 040 - 04F

STATUS: set   FORMAT: II

ACTION: C → Y

PURPOSE: To load the Y register with a constant. Common uses are to set Y to a particular RAM word address, address a selected Digit Latch or to initialize Y for loop control.

DESCRIPTION: The four-bit value from the C-field of the instruction is transferred into the Y register.

MICROINSTRUCTIONS: @CKP, AUTY

3.10.3 Transfer Constant to Memory and Increment Y Register

MNEMONIC: TCMIY C   OPCODE: 060 - 06F

STATUS: set   FORMAT: II

ACTION: C → M(X,Y)
Y + 1 → Y

PURPOSE: Loading the RAM with constants.

DESCRIPTION: The four-bit value from the C-field of the instruction is stored in the memory location addressed by the X and Y registers. The Y register contents are then incremented by one.

MICROINSTRUCTIONS: CKM, @YTP, @CIN, AUTY

3.11 Input/Output Instructions

3.11.1 Reset I/O Pin

MNEMONIC: RSTIO   OPCODE: 00B

STATUS: set   FORMAT: IV

ACTION: 0 → I/O Data Latch I/O Control Latch is set to output direction.

PURPOSE: Output of an active LO at the I/O Pin.

DESCRIPTION: The I/O Data Latch is reset (LO) and the I/O Control Latch is set to output direction. The I/O Pin goes LO in the middle of the instruction.

MICROINSTRUCTIONS: none

3.11.2 Set I/O Pin

MNEMONIC: SETIO OPCODE: 00C

STATUS: set FORMAT: IV

ACTION: 1 → I/O Data Latch
I/O Control Latch is set to output direction.

PURPOSE: Output of an active HI at the I/O Pin.

DESCRIPTION: The I/O Data Latch is set (HI) and the I/O Control Latch is set to output direction. The I/O Pin goes HI in the middle of the instruction.

MICROINSTRUCTIONS: none

3.11.3 Switch I/O Pin to Input

MNEMONIC: IOIN OPCODE: 01A

STATUS: set FORMAT: IV

ACTION: I/O Control Latch is set to input direction.

PURPOSE: Preparation for reads from the I/O Pin.

DESCRIPTION: The I/O Control Latch is set to input direction.
The I/O Data Latch is not affected.

MICROINSTRUCTIONS: none

3.11.4 Transfer Status to I/O Pin

MNEMONIC: TSIO OPCODE: 00F

STATUS: set FORMAT: IV

ACTION: S → I/O Data Latch
I/O Control Latch is set to output direction.

PURPOSE: Conditional output at the I/O Pin.

DESCRIPTION: The I/O Data Latch is loaded with the status. The I/O Control Latch is set to output direction. The I/O Pin settles in the middle of the instruction.

MICROINSTRUCTIONS: none

Example: The memory bit masked by the accumulator is output to the I/O Pin.

```
TCA       8         MASK M(X,Y,3)
ORMNAA              IF M(X,Y,3) = 1: A = 15
ACACC     1         TEST IF A = 15
TSIO                S -> I/O PIN
```

3.11.5 Transfer I/O Pin to Memory

MNEMONIC: TIOM B OPCODE: 0AC - 0AF

STATUS: set FORMAT: III

ACTION: I/O Pin → M(X,Y,B)

PURPOSE: Input instruction for the I/O Pin.

DESCRIPTION: The level at the I/O Pin which was sensed during the previous instruction is transferred to the addressed memory bit.
Note: The I/O Pin is not switched to input direction by this instruction. For I/O input functions the IOIN instruction has to be performed before the TIOM instruction. Otherwise the information of the I/O Data Latch, output by the I/O Pin, is read in.

MICROINSTRUCTIONS: none

Example: Transfer I/O Pin info to M(X,Y,2)

```
IOIN                SWITCH I/O PIN TO INPUT DIRECTION
TIOM      2         I/O INFO -> M(X,Y,2)
```

3.11.6 Transfer Accumulator to K Latch

MNEMONIC: TAK OPCODE: 0B8

STATUS: set FORMAT: IV

ACTION: A → K-Port Latch

PURPOSE: Preparation of output data for the K-Port.

DESCRIPTION: The contents of the accumulator are written into the K-Port Latch. The data appears at the K-Pins if DL8 is set. If DL8 is reset the data is stored only in the K-Port Latch.

MICROINSTRUCTIONS: none

Example: Set K8 and K2 to one, reset K1 and K4 to zero.

```
TCA       10        INFO 8 + 2 -> A
TAK                 MOVE A -> K PORT LATCH
TCY       DL8       SWITCH K PORT TO OUTPUT
SETR                DIRECTION
```

3.11.7 Transfer K Lines to Memory

MNEMONIC: TKM OPCODE: 00A

STATUS: set FORMAT: IV

ACTION: K-Port → M(X,Y)

PURPOSE: Input instruction for the K-Port.

DESCRIPTION: The data at the K-Port is transferred to M(X,Y).

Note: If DL8 is set (K-Port is output) the data from the K-Port Latch is seen at the K-Port lines. Reset DL8 if external data is to be read.

MICROINSTRUCTIONS: CKM

3.11.8 Transfer Data from Accumulator, Status and DL14 to Segment Latches

MNEMONIC: TDO OPCODE: 0B0

STATUS: set FORMAT: IV

ACTION: DL14, S and A decode segment info A to G.
DP Latch defines segment info H
Segment info A to H is transferred into
segment latches decoded by Y register.

PURPOSE: Loading of the display segment buffers.

DESCRIPTION: The actual DL14, status and accumulator contents decode a term out of the 64 term OPLA. The segment A to G info out of this term and the segment H info out of the Decimal Point Latch DP are stored into the segment latches decoded by the Y register.

Note: Which segment info is decoded by DL14, status and accumulator is defined by hardware option .OPTION SEGMENT-PLA (see 2.18.13) Which digit is addressed by the Y register is defined by hardware option .OPTION SELECT-LINES (see 2.18.12)

MICROINSTRUCTIONS: none

3.11.9 Set Decimal Point [Segment H]

MNEMONIC: SDP OPCODE: 0B9

STATUS: set FORMAT: IV

ACTION: 1 → DP

PURPOSE: Activate segment H.

DESCRIPTION: The Decimal Point Latch DP is set. Segment H appears in digits which are loaded by the TDO instruction until DP is reset again.

MICROINSTRUCTIONS: none

3.11.10 Reset Decimal Point [Segment H]

MNEMONIC: RDP OPCODE: 0B6

STATUS: set FORMAT: IV

ACTION: 0 → DP

PURPOSE: Deactivate segment H.

DESCRIPTION: The Decimal Point Latch DP is reset. Segment H disappears in digits which are loaded by the TDO instruction until DP is set again.

MICROINSTRUCTIONS: none

3.11.11 If K Inputs are not equal to Zero, set Status

MNEMONIC: KNEZ OPCODE: 00E

STATUS: comparison result FORMAT: IV

ACTION: 1 → S if K1 + K2 + K4 + K8 # 0
0 → S if K1 + K2 + K4 + K8 = 0

PURPOSE: To test the four K-Port lines for a non-zero state. This instruction is useful for monitoring a keyboard for a "key down" condition.

DESCRIPTION: Data on the four external K-Port lines are compared to zero. Comparison information is transferred into status. Non-zero data inputs cause status to be set.

Note: If DL8 is set (K-Port is output) the data from the K-Port Latch is seen at the K-Port lines. Reset DL8 if external data is to be read.

MICROINSTRUCTIONS: @CKP, NE

3.11.12 Transfer K Inputs to Accumulator

MNEMONIC: TKA OPCODE: 008

STATUS: set FORMAT: IV

ACTION: K-Port → A

PURPOSE: To transfer the K-Port input data into the accumulator for processing.

DESCRIPTION: Data present at the four external K-Port lines is transferred into the accumulator.

Note: If DL8 is set (K-Port is output) the data from the K-Port Latch is seen at the K-Port lines. Reset DL8 if external data is to be read.

MICROINSTRUCTIONS: @CKP, AUTA

3.11.13 Set R Output

MNEMONIC: SETR OPCODE: 00D

STATUS: set FORMAT: IV

ACTION: 1 → DLn

PURPOSE: To set a Digit Latch to a logic one.

DESCRIPTION: The contents of the Y register select the proper Digit Latch. If the contents of the Y register are from 0 to 7 inclusive, outputs R0 to R7 are set. For values greater than 7 the Digit Latches DL8 to DL15 are set.

MICROINSTRUCTION: none

3.11.14 Reset R Output

MNEMONIC: RSTR OPCODE: 036

STATUS: set FORMAT: IV

ACTION: 0 → DLn

PURPOSE: To reset a Digit Latch to a logic zero.

DESCRIPTION: The contents of the Y register select the proper Digit Latch. If the contents of the Y register are from 0 to 7 inclusive, outputs R0 to R7 are reset. For values greater than 7 the Digit Latches DL8 to DL15 are reset.

MICROINSTRUCTION: NONE

3.12 RAM Addressing Instructions

3.12.1 Load X Register with a Constant

MNEMONIC: LDX C OPCODE: 090 - 09F

STATUS: set FORMAT: II

ACTION: C → X
0 → DAM Latch

PURPOSE: Changing the addressed X bank. Resetting the DAM Latch.

DESCRIPTION: The constant from the C field of the instruction is loaded into the X register. The DAM Latch is reset.

Note: The constant C is restricted to the values of the existing RAM banks: 1, 2, 3, 4, 7, 9, 10, 11, 12, 15

MICROINSTRUCTIONS: none

3.12.2 Complement the MSB of the X Register

MNEMONIC:	COMX8	OPCODE: 0B2
STATUS:	set	FORMAT: IV

ACTION: X + 8 → X

PURPOSE: Fast switching from one X bank to another one.

DESCRIPTION: The MSB of the X register is complemented.

MICROINSTRUCTIONS: none

3.12.3 Toggle DAM Latch

MNEMONIC:	TDL	OPCODE: 0BC
STATUS:	set	FORMAT: IV

ACTION: If DAM Latch = 0: 1 → DAM Latch
If DAM Latch = 1: 0 → DAM Latch

PURPOSE: Fast switching from addressed X bank to DAM or reverse. Allows subroutine usage with different X values.

DESCRIPTION: The one bit DAM Latch is toggled. The DAM is addressed if the DAM Latch is set. The X register address is used if the DAM Latch is reset. The contents of the X register are not affected.

Note: The DAM Latch is reset by an LDX instruction.

MICROINSTRUCTIONS: none

3.13 ROM Addressing Instructions

See also chapter 4 SUBROUTINE SOFTWARE

3.13.1 Branch, Conditional on Status

MNEMONIC: BRANCH W OPCODE: 100 - 17F
BR W

STATUS: conditional on Status FORMAT: I

ACTION: If S = 0: PC + 1 → PC
1 → S
If S = 1: W → PC
PB → PA

Note: The PC actually uses a pseudo random count. See Figure 2 for stepping order.

PURPOSE: To allow the program to alter the normal sequential program execution. The BRANCH instruction is conditional on the status results of the previously executed instruction.

DESCRIPTION: The BRANCH instruction is always conditional upon the state of status. If status is reset, then the branch is unsuccessfully executed and the next sequential instruction will be performed. If the status is set, then the branch will occur by the following actions: The W-field of the instruction (7 bits) is transferred to the Program Counter PC. The contents of the Page Buffer PB are transferred to the Page Address PA. After these actions the program continues at the newly loaded page and PC.

Branches may be of 2 types, short or long.

Short branches address within the current page while long branches address into another ROM page. The type of branch performed is determined by the contents of the PB register, which is loaded by the LDP instruction. See 3.13.3 for description of the compound instruction BL (Branch Long).

Note: To allow for conditional branching, the BRANCH instruction must immediately follow the instruction that affected the status. Only that instruction immediately preceding the BRANCH instruction determines if status is zero, causing the branch to be unsuccessfull. If unconditional branching is desired, the preceding instruction must always set status to one.

MICROINSTRUCTIONS: none

3.13.2 Call Subroutine, Conditional on Status

MNEMONIC: CALL W OPCODE: 180 - 1FF

STATUS: conditional on Status FORMAT: I

ACTION:

```
If S = 0:PC + 1 → PC
              1 → S
If S = 1:   SR3 → lost  PSR3 → lost
            SR2 → SR3   PSR2 → PSR3
            SR1 → SR2   PSR1 → PSR2
         PC + 1 → SR1   PA   → PSR1
              W → PC
             PB → PA
```

PURPOSE: To allow the program to transfer control to a common subroutine. Because the CALL instruction saves the return address, subroutines may be called from various locations in a program, and the subroutine will return control back to the proper, saved address after the call using the call-return instruction RETN.

DESCRIPTION: The CALL instruction is always conditional upon the state of status. If status is reset, then the call is executed unsuccessfully and the next sequential instruction will be performed. If the status is set, then the call will occur by the following actions. The address of the next instruction (PC + 1) is pushed onto the Subroutine Return stack. The

Page Address PA is pushed onto the Page Address Return stack. This defines the point where the program will continue after returning from the subroutine.

The stack depth is 3. This allows 3 levels of subroutine nesting. If a 4th level of subroutine nesting is entered, the values stored in SR3 and PSR3 are lost. The W-field of the instruction (7 bits) is transferred to the Program Counter PC. The contents of the Page Buffer PB are transferred to the Page Address PA. After these actions the program continues at the newly loaded page and PC.

Calls may be of 2 types, short or long. Short calls address within the current page while long calls (CALLL) address into another ROM page. The type of call performed is determined by the contents of the PB register, which is loaded by the LDP instruction. See 3.13.4 for description of the compound instruction CALLL.

MICROINSTRUCTIONS: none

3.13.3 Branch Long

MNEMONIC: BL OPCODE: LDP + BRANCH

STATUS: set FORMAT: COMPOUND

ACTION: LDP instruction followed by BRANCH instruction. See description of these instructions for details.

PURPOSE: Branching to a destination outside the current page.

DESCRIPTION: The assembler generates two instructions out of this compound instruction. The correct page is inserted into the LDP instruction and the correct new PC inserted into the BRANCH instruction.

MICROINSTRUCTIONS: none

3.13.4 Call Long Subroutine

MNEMONIC: CALLL OPCODE: LDP + CALL

STATUS: set FORMAT: COMPOUND

ACTION: LDP instruction followed by CALL instruction. See description of these instructions for details.

PURPOSE: Calling a subroutine outside the current page.

DESCRIPTION: The assembler generates two instructions out of this compound instruction. The correct page is inserted into the LDP instruction and the correct new PC inserted into the CALL instruction.

MICROINSTRUCTIONS: none

3.13.5 Load Page Buffer with a Constant

MNEMONIC: LDP C OPCODE: 080 - 08F

STATUS: set FORMAT: II

ACTION: C → PB

PURPOSE: Preparing of destination page for BRANCH or CALL.

DESCRIPTION: The Page Buffer PB is loaded with the four-bit value from the C-field of the instruction. (The ROM page is not changed until the next BRANCH or CALL occurs).

MICROINSTRUCTIONS: none

3.13.6 Return from Subroutine

MNEMONIC: RETN OPCODE: 0BF

STATUS: set FORMAT: IV

ACTION: 0 → ADD Latch
0 → BRANCH Latch

If inside Subroutine: SR1 → PC PSR1 → PA
PSR1 → PB
SR2 → SR1 PSR2 → PSR1
SR3 → SR2 PSR3 → PSR2
If not inside Subroutine: No further action

PURPOSE: Returning from a subroutine.

DESCRIPTION: The ADD Latch and BRANCH Latch are reset. If the program is inside a subroutine the Program Counter and the Page Address register are loaded from the top of the subroutine stack. Then the stack is raised one level up, leaving the next return address on top of the subroutine stack. If the program is not inside a subroutine, no further action occurs.

MICROINSTRUCTIONS: none

3.14 Analog-Digital-Converter Instructions

3.14.1 Set Converter Supply

MNEMONIC: SCSV OPCODE: 0BA

STATUS: set FORMAT: IV

ACTION: Pin SV_{DD} is switched on.
Comparator offset switch → NONINVERTED

PURPOSE: Supplying the ADC part and sensors with voltage.

DESCRIPTION: The supply converter voltage is switched to the external sensors and the resistor network of the comparator. The comparator offset switch is switched to "NONINVERTED".

MICROINSTRUCTIONS: none

3.14.2 Reset Converter Supply Voltage

MNEMONIC: RCSV OPCODE: 0BB

STATUS: set FORMAT: IV

ACTION: Pin SV_{DD} is switched off.

PURPOSE: Switching off the ADC part.

DESCRIPTION: The supply converter voltage SV_{DD} is switched off.

Note: The supply converter voltage SV_{DD} is switched off too by initialization and the instructions DONE and OFF.

MICROINSTRUCTIONS: none

3.14.3 Reset Comparator Offset switch to Inverted

MNEMONIC: RCI OPCODE: 0BD

STATUS: set FORMAT: IV

ACTION: Comparator offset switch → INVERTED.

PURPOSE: To initialize measurement with inverted comparator inputs. The addition of a measurement result with inverted comparator and one with noninverted comparator compensates the offset error of the comparator.

Note: The comparator offset switch can be reset to the NONINVERTED state by the instruction SCSV.

DESCRIPTION: The comparator offset switch is set to INVERTED.

MICROINSTRUCTIONS: none

3.14.4 Transfer Comparator Output to Memory Bit

MNEMONIC: TCTM B OPCODE: 024 - 027

STATUS: set FORMAT: III

ACTION: Comparator output → M(X,Y,B)
1 → M(X,Y,B-1)

PURPOSE: Building the ADC value by successive approximation.

DESCRIPTION: The comparator output is written to the addressed memory bit M(X,Y,B). Then the next lower bit of the addressed nibble is set to one to prepare for the next comparison. If the LSB of the nibble is addressed, no additional bit is set.

Note: For A/D conversions the addressed memory has to be M(15,13-15). Only these RAM nibbles are connected to the A/D converter for comparison.

MICROINSTRUCTIONS: none

3.14.5 Transfer Range to Memory Bit

MNEMONIC: TRTM B OPCODE: 028 - 02B

STATUS: set FORMAT: III

ACTION: If M(15,13-15) = >000: 1 → M(X,Y,B)
If M(15,13-15) = >FFF: 1 → M(X,Y,B)
Otherwise : 0 → M(X,Y,B)

PURPOSE: Fast validity checking of the measured ADC value.

DESCRIPTION: The range output is written to the memory bit addressed by the operand field. If the measured value is inside the voltage range of the ADC the bit is set to zero. If the measured voltage is below the voltage range of the ADC (ADC value is 000) or above the voltage range of the ADC (ADC value is >FFF) the addressed bit is set to one.

Note: The TRTM instruction checks only the contents of the RAM nibbles M(15,13-15) independent if the stored values came from an ADC conversion or not.

MICROINSTRUCTIONS: none

3.15 Miscellaneous Instructions

3.15.1 Enter Done Mode

MNEMONIC: DONE OPCODE: 0BE

STATUS: not applicable FORMAT: IV

ACTION: CPU part is switched off. SV_{DD} is switched off.

PURPOSE: Current saving by deactivation of unnecessary parts.

DESCRIPTION: The CPU part is switched off, only the RAM, the LCD drive, the timer and other parts stay active. See Figure 6 for more information. A delay line is read out which wakes up the CPU if during the last 5 instructions before the DONE instructing an external event occured which would wake up the processor during the Done Mode. This ensures that every event can be recognized. See Figure 5 for the conditions. The ADC supply voltage SV_{DD} is switched off too.

MICROINSTRUCTIONS: none

3.15.2 Enter Off Mode

MNEMONIC: OFF OPCODE: 0B7

STATUS: not applicable FORMAT: IV

ACTION: CPU, SV_{DD} and timer are switched off.

PURPOSE: Current saving by deactivation of not necessary parts.

DESCRIPTION: The CPU and the timer are switched off, only the RAM stays active. Events at the inputs which would wake up the CPU from Done Mode, will wake up the CPU from this mode too. See Figure 5 for the conditions. The ADC supply voltage SV_{DD} is switched off too.

MICROINSTRUCTIONS: none

3.16 Timer Instructions

3.16.1 Transfer Timer to Accumulator

MNEMONIC: TTA OPCODE: 01E

STATUS: set FORMAT: IV

ACTION: If DL12 = 0: Timer0 → A
If DL12 = 1: Timer1 → A

PURPOSE: Reading of the timer.

DESCRIPTION: The timer nibble which is addressed by DL12 is transferred to the accumulator.

Note: A check is necessary to assure a stable value. This avoids reading temporary timer values.

MICROINSTRUCTIONS: @CKP, AUTA

Example: The timer is read into the accumulator. To be shure that a stable value was read in, the timer is read in once more if the value in the accumulator differs from the timer value.

```
L$1        TTA                          TIMER -> A
           TNEA                         YET SAME VALUE ?
           BRANCH     L$1               IF NOT READ ONCE MORE
           ...                          STABLE VALUE IN A
```

3.16.2 Transfer Timer to Y Register

MNEMONIC: TTY OPCODE: 01F

STATUS: set FORMAT: IV

ACTION: If DL12 = 0: Timer0 → Y
If DL12 = 1: Timer1 → Y

PURPOSE: Reading of the timer.

DESCRIPTION: The timer nibble which is addressed by DL12 is transferred to the Y register.

Note: A check is necessary to assure a stable value. This avoids reading temporary timer values.

MICROINSTRUCTIONS: @CKP, AUTY

Example: The timer info read into the Y register is compared with a second read into the accumulator. If the info differs, the operation is repeated once more.

```
L$500     TTY                     TIMER -> Y
          TTA                     TIMER -> A
          YNEA                    SAME INFO ?
          BRANCH    L$500         NO, READ ONCE MORE
          ...                     YES, PROCEED
```

3.16.3 Transfer Timer to Memory

MNEMONIC: TTM OPCODE: 01B

STATUS: set FORMAT: IV

ACTION: If DL12 = 0: Timer0 → M(X,Y)
If DL12 = 1: Timer1 → M(X,Y)

PURPOSE: Reading of the timer.

DESCRIPTION: The timer nibble which is addressed by DL12 is transferred to the addressed memory.

Note: A check is necessary to assure a stable timer value. This avoids reading temporary values.

MICROINSTRUCTIONS: CKM

Example: The timer is read into the addressed memory. To be shure that a stable value was read in the timer is read in once more if the value in M(X,Y) differs from the timer value.

```
L$100     TTM                     TIMER -> M(X,Y)
          TNEM                    YET SAME VALUE ?
          BRANCH    L$100         IF NOT READ ONCE MORE
          ...                     STABLE VALUE IN M(X,Y)
```

3.16.4 If Timer not equal to Memory, One to Status

MNEMONIC: TNEM OPCODE: 01C

STATUS: comparison result FORMAT: IV

ACTION:
If DL12 = 0:
If Timer0 # M(X,Y): 1 → S
If Timer0 = M(X,Y): 0 → S
If DL12 = 1:
If Timer1 # M(X,Y): 1 → S
If Timer1 = M(X,Y): 0 → S

PURPOSE: Assuring a stable timer readout.

DESCRIPTION: The timer nibble which is addressed by DL12 is compared against the addressed memory M(X,Y). If the contents are equal, status is reset. If they are not equal, status is set. See example at 3.16.3.

MICROINSTRUCTIONS: @CKN, @MTP, NE

3.16.5 If Timer not equal to Accumulator, One to Status

MNEMONIC: TNEA OPCODE: 01D

STATUS: comparison result FORMAT: IV

ACTION:
If DL12 = 0:
If Timer0 # A: 1 → S
If Timer0 = A: 0 → S
If DL12 = 1:
If Timer1 # A: 1 → S
If Timer1 = A: 0 → S

PURPOSE: Assuring a stable timer readout.

DESCRIPTION: The timer nibble which is addressed by DL12 is compared against the accumulator. If the contents are equal, status is reset. If they are not equal, status is set. See example at 3.16.1.

MICROINSTRUCTIONS: @ATN, @CKP, NE

4 Subroutine Software

4.1 General

By using the subroutine capability of the TSS400(/4) series, programs are substantially compacted, enabling the user to write very powerful algorithms within the 2048 word limit. Normally "straight line" programming is used only for high speed applications.

A subroutine is used to avoid duplication of ROM code when a particular section of code is used several times within a program. A subroutine is a section of code terminated with a RETN instruction. A CALL instruction transfers program execution to the first instruction in the subroutine. At the completion of the subroutine, program control is transferred to the instruction address immediately following the CALL instruction. Examples of a subroutine and different calling techniques follow.

4.2 Nesting Subroutines

The TSS400(/4) has a subroutine stack with a depth of 3. This means that 3 levels of subroutines are possible (a subroutine can call a subroutine and this one can call a subroutine too). This procedure is called nesting subroutines. If a 4th level of subroutines is used, the return information of the 1st level is lost and the software will not work properly. So the programmer is advised always to be informed which subroutine level he is using when coding programs.

Example: The following program shows subroutine nesting 3 levels deep.

```
MAINLOOP  CALLL     TEST      INVOKE TEST SUBROUTINE
          ...                 FROM MAINLOOP: 1st LEVEL
*---------
TEST      CALLL     PREPARE   CALL 2nd LEVEL
          CALLL     DISPLAY   CALL 2nd LEVEL AGAIN
          RETN
*
PREPARE   CALL      CLRDISPL  CALL 3rd LEVEL
          TCY       YMSD
          TCMIY     1
          ...
          RETN
*
CLRDISPL  LDX       XDISPL    ROUTINE CLEARS DISPLAY BUFFER
          TCY       0         M(XDISPL,0-8)
          TCMIY     BLANK
          ...
          RETN
```

4.3 Example Calling Sequence

The following subroutine CREG will clear digits 0 to 6 on a given RAM bank. The X register value is set prior to the CALL.

```
         ORG       896       PLACE CREG ON PAGE 7 PC 00
*                            (7x128 = 896)
CREG     TCY       0         INITIALIZE Y TO 0
C1       TCMIY     0         CLEAR NIBBLE
         YNEC      7         CONTINUE UNTIL NIBBLE 6 IS 0
         BRANCH    C1        WHEN Y = 7, RETURN TO THE CALLING
         RETN                PROGRAM
```

Recall that both CALL and BRANCH instructions are conditional on status. To successfully execute a call, the CALL instruction must either follow a non-status affecting instruction, such as TCY or LDX, or follow a status affecting instruction that will leave status set at one. Care must be taken to ensure that the Page Buffer contents point to the subroutine page before attempting a call. This is always sure if the CALLL directive is used which inserts the proper LDP instruction automatically. The subroutine labeled CREG is arbitrarily placed on ROM page 7 PC 00. To demonstrate calling sequences, examples of calling from page 7 and calling from other pages are given.

4.3.1 Calling a Subroutine on the same page

Remember that a successful branch to a page transfers the Page Buffer contents to the Page Address register, leaving them identical. The first example assumes that both registers are identical. By preceding the call with a non-status affecting instruction, the call will be unconditional.

```
          LDX       3         STATUS = 1 AFTER LDX
          CALL      CREG      SO CREG IS ALWAYS CALLED
          BR        XYZ       M(3,0-6) IS CLEARED
```

When a subroutine is used initially, it is embedded directly in a straight line sequence rather than called, to save a CALL instruction. This holds only if the Page Buffer and Page Address register contents are identical.

```
          LDX       3         PB = PA = 7
CREG      TCY       0         PASS THROUGH CREG
C1        TCMIY     0
          YNEC      7
          BRANCH    C1
          RETN                NOP WHEN NOT "CALLED"
          BR        XYZ
```

The following example shows how to correctly call a subroutine when the Page Buffer has been modified. If accumulator is less or equal to 6 a branch to label ABC located on page 4 should be made. If this condition is not met the subroutine CREG located on the current page 7 is to be called.

```
          LDP       4         PREPARE PB FOR BRANCHING TO PAGE 4
          ALEC      6         IF A IS LESS THAN 7, BRANCH TO ABC
          BRANCH    ABC
          LDP       7         CHANGE PAGE BUFFER BACK TO 7
          CALL      CREG      CALL CREG CORRECT ON CURRENT PAGE
          BR        LABEL
```

The above mentioned example is for demonstration purposes only. The LDP instruction should never be used explicitly but only implicitly with the directives BL and CALLL. These directives insert the right page into the LDP instruction automatically. This avoids branches to wrong pages if the addressed labels are shifted to other pages and the LDP constant wasn't corrected. The better sequence for the above example would be:

```
          ALEC      6         IF A IS LESS THAN 7, BRANCH TO ABC
          BRANCH    PABC
          CALL      CREG      CALL CREG CORRECT ON CURRENT PAGE
          BR        LABEL     AND GO ON
PABC      BL        ABC       ASSEMBLER INSERTS LDP 4 AND BR ABC
```

The LDP instruction should only be used explicitly if clarity is better than compared with BL. An example is a branch table with all destinations located on the same page 8. Branching depends on accumulator contents:

```
LDP       8         PREPARE PAGE BUFFER
ALEC      0         LABELS CODE0 TO CODE15 IN PAGE 8
BRANCH    CODE0     AC = 0
ALEC      1
BRANCH    CODE1
...
...
ALEC      14
BRANCH    CODE14
BR        CODE15    AC = 15
```

4.3.2 Calling a Subroutine from a different page

If the calling sequence is not on the same page as the subroutine, a LDP precedes the CALL so that the Page Buffer contents are equal to the subroutine's page address. This is accomplished normally by using the CALLL directive:

```
LDX       2         CLEAR M(2,0-6)
CALLL     CREG
LDX       3         CLEAR M(3,0-6)
CALLL     CREG
```

The following example demonstrates a CALL from one page to another with a conditional CALL. Current page is 4.

```
LDP       7         SET PAGE BUFFER TO 7
ALEC      0         IF A = 0, THEN CREG WILL BE
CALL      CREG      SUCCESSFULLY CALLED.
LDP       4         SET PAGE BUFFER BACK TO 4
BR        XYZ       PROCEED AT PAGE 4
```

Notice that the test instruction, ALEC 0, must immediately precede the CALL since status is affected for one instruction cycle only. If label XYZ is located on page 7 too, the LDP 4 instruction is to be ommitted.

Note: As mentioned at 4.3.1 the last example is only for demonstration purposes. The correct way for coding would be as follows:

```
          ALEC      0         IF A = 0, THEN CREG WILL BE
          BRANCH    PCREG     CALLED BY CALLL
          BR        XYZ       A # 0 PROCEED
PCREG     CALLL     CREG      POINTER TO CREG
          BR        XYZ
```

4.4 Multiple Entry Points

Often it is desired to use a subroutine several times, specifying different conditions each time for entering that subroutine. A call to the multiple entry points presets different conditions, and then a branch into the base subroutine is executed. Thus, rewriting the subroutine for each entry condition is avoided.

The following examples use the CREG routine as the basic subroutine. The CREG routine clears words 0 to 6 on a RAM bank where the X address is set before CREG is called (as in example, paragraph 4.3.1). If clearing words 0 to 6 is required more than once in a program, then it is advantageous to create a new subroutine (e.g., CREG3 which sets X to 3 before entering CREG).

```
CREG15    LDX       15        CLEAR M(15,0-6)
          BR        CREG
*
CREG3     LDX       3         CLEAR M(3,0-6) AND GO TO CREG
*
CREG      TCY       0         CLEAR M(X,0-6)
C1        TCMIY     0
          YNEC      7         BASIC SUBROUTINE
          BRANCH    C1
          RETN
```

Now the calling sequence for clearing M(15,0-6), M(3,0-6) and M(2,0-6) becomes:

```
          CALL      CREG15    CLEAR M(15,0-6)
          CALL      CREG3     CLEAR M(3,0-6)
          LDX       2
          CALL      CREG      CLEAR M(2,0-6)
```

Note: that the CREG subroutine is not modified and can be called again (like explained above for X = 2)

Another example, again using CREG as the base subroutine, is the subroutine CLALL. CLALL sets Y to 7 before entering the clearing routine, so every word on the RAM bank is cleared, including words 7 to 15.

```
CLALL     TCY       7        SET Y = 7. CLEAR M(X,0-15)
          BR        C1       BRANCH INTO THE CLEARING LOOP
*
CREG      TCY       0
C1        TCMIY     0        ENTRY POINT FOR CLALL
          YNEC      7        BASIC SUBROUTINE
          BRANCH    C1
          RETN
```

Generally, less LDP instructions are needed when a subroutine resides on the same ROM page that the subroutine calls most frequently.

5 Organizing the RAM

5.1 General

To use the TSS400(/4) data storage efficiently, the locations of storage areas in the RAM must be assigned carefully. The RAM is normally subdivided into data "registers", flag bits, and temporary working areas. If the location assignments are chosen logically, unnecessary use of RAM addressing instructions, such as LDX and TCY, can be minimized.

The following paragraphs give general guidelines for RAM organization which are useful in most programs.

Organization of the RAM should start with organizing the DAM.

This results from the special assignment of the DAM for A/D conversion and computing with Special Arithmetic Instructions:

- DAM locations M(15,13-15) are hardwired to the Digital Analog Converter which controls the A/D conversion. This dictates usage for the ADC if it is used in the application.
- The Special Arithmetic Instructions work only together with the DAM, so if fast arithmetic is needed (e.g. the Integer Math. Package shown under 6.5) one computation register (REGB) is to be located in the DAM.

Example: The ADC is used and a 9 digit Integer Math. Package is necessary. The resulting DAM would look like as follows:

← Data Register REGB →													← ADC →			X = 15
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Y
MSD								LSD	+/-				MSD		LSD	

5.2 Data Register Organization

To minimize X and Y addressing, data "registers" should be located on sequential locations on the same RAM bank. For example, the following is a seven word "register" organization that defines a subset of any 16 word bank:

<table>
<tr><td colspan="7">← Data Register →</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>Y</td></tr>
</table>

If organized in this manner, a register left-shift subroutine requires only three Y addressing instructions.

Note: Assume in this example and in all succeeding examples, that register location Y = 0 is the most significant digit (MSD) and location Y = 6 is the least significant digit (LSD).

5.2.1 Register left Shift Example

The subroutine LSHIFT multplies the register M(X,0-6) by ten if decimal numbers are used. If another numbering base is used the register is multiplied by this base. For example: 0123456 → 1234560

```
LSHIFT     CLA                    ENTER WITH SET X
LDATA      TCY        6           THE LOCATION Y = 6 IS THE LSD.
L1         XMA                    EXCHANGE MEMORY AND ACCUMULATOR.
          DYN                    TO NEXT HIGHER DIGIT
          BRANCH     L1
          RETN
```

Note that this subroutine could also be a data entry subroutine by setting the accumulator equal to the new LSD data and calling DATA.

```
          TCA        5          XXXXX -> XXXXX5
          CALL       LDATA      USE 2nd ENTRY POINT
          . . .
```

For most programs, several data registers are required. Whenever possible, these registers are placed on different RAM banks, and their Y locations on these banks would be equal. Thus, X and Y addressing can be minimized on register-to- register operations such as transfers, addition and subtraction. If organized as in the figure below, a transfer from DR9 to register DR1 requires only three addressing instructions.

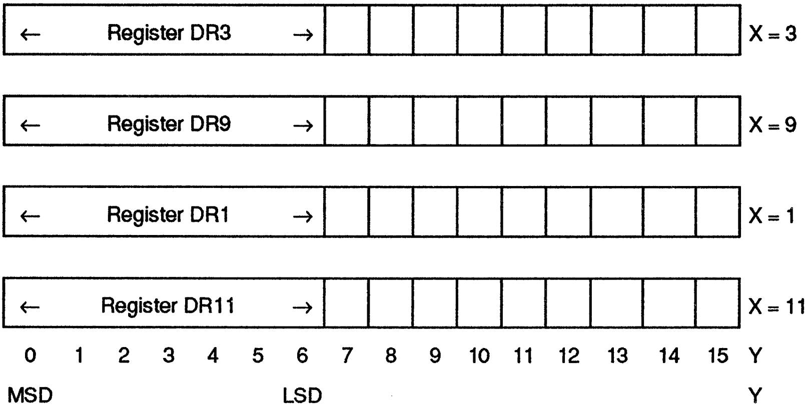


Figure 18: Register Example

5.2.2 Transfer from Register 9 to 1 [Example]

```
TR91      TCY       6       INITIALIZE Y = 6 (LSD)
T1        LDX       9       SET X = 9
          TMA               TRANSFER M(9,Y) TO ACCUMULATOR
          LDX       1       SET X = 1
          TAMDYN            TRANSFER ACCUMULATOR TO M(1,Y) AND
          BRANCH    T1      DECREMENT Y UNTIL Y = 15
          RETN
```

5.2.3 Register Transfer Example using COMX8

Notice in paragraph 5.2.2 that DR9 contents can be transferred to DR1, but DR1 cannot be transferred to DR9. Also, this subroutine cannot be used for transfers between any other register pairs.

This limitation can be overcome by using the COMX8 instruction in the following subroutine and by defining paired registers (two registers that transfer to and from each other) to be on X banks with complementary MSBs.

```
TRB3      LDX       3       M(11,0-6) -> M(3,0-6)
*                           BASE SUBROUTINE FOR ALL OTHERS:
TR0       TCY       6       INITIALIZE Y = 6 (LSD)
```

```
T2          COMX8                   COMPLEMENT MSB OF X
            TMA                     TRANSFER M(X,Y) TO ACCUMULATOR
            COMX8                   COMPLEMENT MSB OF X
            TAMDYN                  TRANSFER ACCUMULATOR TO M(X,Y),
            BRANCH      T2
            RETN
*
* OTHER ENTRY POINTS FOR THE ABOVE SUBROUTINE:
*
TR3B        LDX         11          M(3,0-6) -> M(11,0-6)
            BR          TR0
*
TR19        LDX         9           M(1,0-6) -> M(9,0-6)
            BR          TR0
*
TR91        LDX         1           M(9,0-6) -> M(1,0-6)
            BR          TR0
```

By using multiple entry points, four register transfers are accomplished with one base subroutine.

5.3 Placing FLAG Bits

One should carefully choose the location of flag bits. As usual, the objective is to minimize addressing instructions. The following are general suggestions on bit placement.

Registers and the registers' sign bits should be located in the same RAM bank and located in adjacent Y addresses. This permits the transfer of sign bits by the same subroutine which transfers the register contents. For instance, in examples in paragraphs 5.2.2 and 5.2.3, sign bits located in Y = 7 could be transferred, along with the data in the register, simply by changing the initializing TCY 6 to a TCY 7 command.

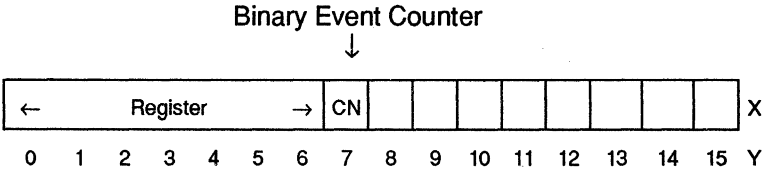
Different flags which are tested sequentially in a program should be placed in the same RAM word to eliminate both X and Y changes between tests.

For flags which are used with several X banks the placement in the DAM should be considered. See 5.5 for examples.

5.4 Temporary Working Areas

Temporary working areas are either full register length, which is required in a three register calculation (i.e., divide, when one register holds the intermediate results), or shorter length areas which are used for counters and pointers.

Pointers and counters should be placed on the same RAM bank as the registers that they interact with. The following example monitors an external event (detected at EXIT label), counts to 16 (e.g., 16 items loaded into a container), and then adds one to a BCD register in the same bank for each count of 16.



```
COUNT      TCY        7          ADDRESS EVENT COUNTER
           IMAC                  COUNTER + 1
           BRANCH     ADD1       IF CARRY, ADD 1 TO REGISTER
           TAM                   IF NO CARRY A -> M(X,7)
           BR         EXIT       WAIT FOR NEXT EVENT
*
ADD1       TAMDYN                16'TH PULSE: 0 -> M(X,7)
INCM       IMAC                  M(X,Y) + 1 -> A
           TAM                   A ->  M(X,Y)
           ALEC   9              BCD CORRECTION NECCESSARY ?
           BRANCH     EXIT       NO, WAIT FOR NEXT EVENT
           TCMIY      0          YES, 0 -> M(X,Y)
           ACYY       14         ADDRESS NEXT HIGHER DIGIT
           BR         INCM       AND INCREMENT IT
*
EXIT       .                     EXTERNAL ROUTINE WAITS FOR THE
           .                     COMPLETION OF A NEW EVENT THAT IS
           .                     TO BE COUNTED.
           BR         COUNT      EVENT OCCURED, INCR. EVENT CTR.
```

Notice that by placing the BCD counter and event counter in the same bank, no X addressing was required. The LSD of the BCD register was addressed with TAMDYN, saving both ROM code and execution time.

5.5 DAM Addressing

Three RAM addressing instructions allow to address the DAM immediately without changing the contents of the X register.

The above mentioned instructions are:

TDL Toggle DAM Latch
TDA Transfer addressed DAM nibble to accumulator
XDA Exchange addressed DAM nibble and accumulator

This saves ROM code due to 2 reasons:
- The X register can stay unchanged despite referencing of another X register (the DAM)
- One properly written subroutine can be used with every X bank despite referencing the DAM

Example: The checksum for an X bank is to be computed and stored in M(15,0) and accumulator. The X register is loaded before the call.

```
          LDX       3          ADDRESS X-BANK
          CALL      CHKSUM     CALL CHECKSUM SUBROUTINE
          ...                  M(3,0) IS ADDRESSED

CHKSUM    CLA                  CLEAR SUMMING REGISTER
          TCY       15         START WITH M(X,15)
L$105     AMAAC                ADD M(X,Y) TO A
          DYN
          BRANCH    L$105      M(X,15-0) ADDED UP ?
*
          TCY       0          YES, CHECKSUM IN A
          XDA                  CHECKSUM -> M(15,0)
          TDA                  M(15,0) -> A
          RETN                 M(X,0) IS ADDRESSED
```

Often referenced flags may be placed in the DAM where they can be tested without changing the X register. They are located best between the Integer Math Register REGB and the locations used by the ADC M(15,13-15).

Example: Dependent on FLAG1 located in M(15, FLAGS, FLAG1) the program should take different pathes.

```
          LDX       3
          TCY       7
          TCMIY     15          15 -> M(3,7)
          TCY       FLAGS       FETCH FLAGS FOR TEST
          TDA                   M(15,FLAGS) -> A
          TBITA     FLAG1       FLAG1 SET ?
          BRANCH    FLAG1SET    YES, M(3,FLAGS) ADDRESSED
          BR        FLAG1CLR    NO, M(3,FLAGS) ADDRESSED
```

If neither the contents of the X register nor the contents of the Y register may be lost, despite information has to be fetched from the DAM, the following example may be of help.

Example: The Information of M(15,FLAGS) is needed with several X and Y register contents which may not be lost. If FLAG2 located in M(15,FLAGS) is set, a subroutine SUBR is to be called which processes data in the previous addressed RAM nibble. A temporary storage nibble M(15,TEMP) is needed for the Y register.

```
          LDX       4
          TCY       8           M(15,FLAGS) INFO NEEDED IN A
          CALL      CHKFLAGS    M(15,FLAGS) -> A
          TBITA     FLAG2       M(4,8) IS STILL ADDRESSED
          CALL      SUBR        CALL SUBR IF FLAG2 IS SET
          ...                   SUBR WORKS ON M(4,8)
          ...
CHKFLAGS  TYA                   STORE Y IN A
          TDL                   ADDRESS DAM
          TCY       TEMP        ADDRESS M(15,TEMP)
          TAM                   STORE Y
          TCY       FLAGS
          TMA                   FLAGS M(15,FLAGS) -> A
          TCY       TEMP
```

```
TMY                     RESTORE Y
TDL                     ADDRESS M(X,Y) AGAIN
RETN                    M(X,Y), FLAGS IN A
```

General the usage of the 3 direct DAM instructions can be described as follows:

TDL Change of the addressed RAM to the DAM if more complex modifications than transfers are necessary. A second TDL instruction addresses the previous X register again.

TDA Transfers to the DAM

XDA Transfers from the DAM. Note that the accumulator is replaced by the contents of M(15,Y). If the transferred information is still needed in the accumulator, a TDA instruction after the XDA instruction may be used. See first example above.

6 General Purpose Subroutines

This chapter shows and comments some useful subroutines which have been tested thouroughly in several applications. Some subroutines call other subroutines: if this happens, the called subroutine is shown also.

The TSS400(/4) assembler knows both mnemonics of the BRANCH instruction (BR, BRANCH). It is good practice to use the two mnemonics in different ways despite the fact that they share the same opcode.

BR	Status is always 1 due to the previous instruction
BRANCH	Branch depends on status of previous instruction

Another good programming practice is the use of "Local Assembly Labels". This means that labels, which are referenced only inside the subroutine where they are defined, have names in the form L$xxx, where xxx is a unique decimal number. Otherwise labels which are entry points should get labels which give a certain description e.g. DIVISION. This simplifies the distinction between outside referenced labels and local labels.

The software examples all use the above mentioned nomenclature whenever possible.

6.1 Register Shift Right [Rounding Routine]

This routine is used for dividing a BCD register like the FLAC by 10. If the LSD is less than four, only shifting right takes place, if it is five or greater, the LSD+1 is incremented (rounded up) and the result of the complete register corrected if necessary. Any register arranged like the FLAC can be shifted and rounded if the second entry "ROUND" is used and the X register is loaded appropriate prior to the CALL.

Example: If the Help Register M(RRXHR,0-RRVZ) is to be rounded the following call sequence can be used:

```
LDX     RRXHR   LOAD APPROPRIATE X. SHIFT
CALL    ROUND   HELP REGISTER M(RRXHR,0-RRVZ)
```

If frequently more than one shift is necessary, which means divisions by 100, 1000 or 10000, a subroutine can save ROM by calling the shift routine more than once. Three entries are shown, note that 2 levels of subroutine nesting are used. The subroutine "falls through" to the "ROUNDFL" subroutine which saves 2 instructions.

```
DIV10E4    CALL      ROUNDFL   DIVIDE FLAC BY 10000
DIV10E3    CALL      ROUNDFL   DIVIDE FLAC BY 1000
DIV10E2    CALL      ROUNDFL   DIVIDE FLAC BY 100
*-------------------------------------------------------------
*
* ROUTINE DIVIDES FLAC M(RRXFL,0-RRLSD) BY 10 AND
* ROUNDS UP IF LSD > 4.
*
ROUNDFL    LDX       RRXFL     ADDRESS LSD M(RRXFL,RRLSD)
ROUND      TCY       RRLSD     ENTRY FOR SHIFTING OTHER REG.
           SAL                 SET ADD LATCH
           REAC                SS = 0
           TMA
           ALEC      4         IS LSD < 5 ?
           BRANCH    L$000     YES, DO NOT ROUND UP
           SEAC                NO, SET SS FOR ROUNDING UP
L$000      DYN                 ADDRESS NEXT HIGHER DIGIT
           CCLA                SS -> AC
           AMAAC               ADD DIGIT TO AC (0 OR 1)
           IYC
           TAMACS    6         TRANSFER TO NEXT LOWER DIGIT
           CTMDYN              AND CORRECT IF NECESSARY
           YNEC      RRMSD     MSD M(RRXFL,0) REACHED ?
           BRANCH    L$000
           CCLA                YES, STORE OVERFLOW OUT OF
           TAM                 MSD-1 INTO MSD
           RETN
```

6.2 Register Clear Routine

Before registers are loaded with new constants, they should be cleared (set to +0). The X register is loaded with the right value before branching to the main part label CLR.

```
CLRZWSP    LDX       XZWSP     CLEAR INTERMEDIATE BUFFER
           BR        CLR       USE 2nd ENTRY POINT
CLRREGB    LDX       RRXRB     CLEAR REGB M(RRXRB,0-RRVZ)
           BR        CLR       USE 2nd ENTRY POINT
*
CLRFLAC    LDX       RRXFL     CLEAR FLAC M(RRXFL,0-RRVZ)
```

```
CLR       TCY       RRMSD     2nd ENTRY POINT
L$001     TCMIY     0
          YNEC      RRVZ+1    SIGN CLEARED TOO ?
          BRANCH    L$001     NO, ONCE MORE
          RETN
```

6.3 Display Driver Routine

The display driver routine below outputs a display buffer located in M(XDISPL,0-4). Digits in Y = 0 to 3 contain numbers or blanks while M(XDISPL,4) contains flags which are output binary. The same Y register contents address the RAM and the Select Lines. If this is not true, an additional TCY n instruction is necessary after the TMA instruction for selecting the correct Select Lines.

The subroutine assumes the following conditions:

OPLA definition for numbers: DL14 = 0, status = 1
OPLA definition for flags: DL14 = 0, status = 0

```
DISPLAY   CALL      ZEROSUPP  SUPPRESS LEADING ZEROES
          TCY       DL14      X = XDISPL
          RSTR                SWITCH TO OPLA PART DL14 = 0
          RDP                 OUTPUT WITHOUT SEGMENT H
          TCY       0         START WITH MSD
          TMA
          TDO                 OUTPUT TO DIGIT 0
*
          TCY       1         MSD-1
          TMA
          TDO                 DIGIT 1
*
          TCY       2         LSD+1
          TMA
          SDP                 SET DEC. POINT
          TDO
*
          TCY       3         LSD
          TMA
          RDP
          TDO                 S11+S12
*
          TCY       4         FLAGS IN DISPLAY
          TMA
          ACACC     0          RESET STATUS FOR BINARY OUTPUT
          TDO
          RETN
```

6.4 Leading Zero Suppression

For good readability of a display it is normally necessary to suppress leading zeroes. The following subroutine suppresses all leading zeroes with exception of the least significant digit. The leading zeroes are replaced by the code which is defined as blank character in the OPLA.

```
ZEROSUPP  LDX       XDISPL     START WITH THE MSD OF THE
          TCY       YMSD       DISPLAY BUFFER M(XDISPL,YMSD-YLSD)
L$002     MNEZ                 IS CURRENT DIGIT 0 ?
          BRANCH    L$003      NO, RETURN TO MAIN PROGRAM
          TCMIY     BLANK      YES, REPLACE IT WITH BLANK
          YNEC      YLSD       LSD REACHED ?
          BRANCH    L$002      NO, TEST NEXT LOWER DIGIT
L$003     RETN                 YES, DON'T MODIFY IT
```

6.5 Integer Math Package

This package consists of four subroutines for signed addition, subtraction, multiplication and division. The sign is located in M(X, RRVZ, 3).

Three RAM registers are used by the package:

FLAC: holds 1st operand and result after operation

REGB: holds 2nd operand (not modified by operation) exception: subtraction changes the sign of REGB).

HELP: Used as a help register for multiplication and division (not necessary if neither multiplication nor division is used)

The following register example shows the RAM like defined in the shown Integer Math Package:

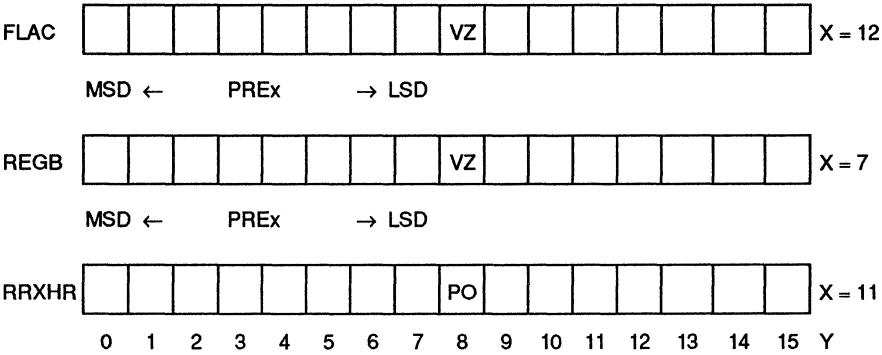


Figure 19: Register Example for the Integer Math Package

Name	Subroutine Call	Operation	Used ROM
Addition	CALL ADDITION	FLAC = FLAC + REGB	48
Subtraction	CALL SUBTRACT	FLAC = FLAC - REGB	4
Multiplication	CALL MULTIPLY	FLAC = FLAC · REGB	40
Division	CALL DIVISION	FLAC = FLAC : REGB	64

The formulas for the execution times of the four routines can be found at the beginning of the software package. Before calling one of the four subroutines, the registers FLAC and REGB have to be loaded with the numbers to be processed. The result of the operation is transferred to the FLAC register.

The ADDITION subroutine can be used for all X banks if the second entry ADDREGX is used:

```
LDX     X          M(X,0-RRLSD) + REGB ->
CALL    ADDREGX    M(X,0-RRLSD)
```

The register length can be tailored to the needed accuracy. The length can range from 2 to 15 digits. One nibble is used for the sign resp. the pointer of the register. The routines are designed for a maximum of speed and a minimum of ROM space. This is accomplished by use of the Special Arithmetic Instructions explained in 3.6.

The definitions for the package can be modified as follows:

```
RRXFL     1, 2, 3, 4, 9, 10, 11, 12   X FLAC
RRLSD     1 - 14                      Y LSD
RRVZ      2 - 15                      Y SIGN (RRLSD+1 !)
RRXHR     1, 2, 3, 4, 9, 10, 11, 12   X HELP (RRXFL # RRXHR)

RREx      means 10Ex digit
```

All other definitions of the package are fixed and may not be modified!

The package is an Integer Package which means that the position of the decimal point has to be controlled by the programmer. To show where the decimal point is assumed, one should note the format of the number at the right margin of the source as it is done in the following examples:

XXX.YY	two digits right of the thought decimal point
XXXXXX	integer number without fraction

The digit count of "X" can be used to show the largest possible number of digits left of the decimal point.

The rules for the position of the decimal point are:

- Addition: Positions after the decimal point have to be equal. Position is the same for the result.
- Subtraction: Same as addition.
- Multiplication: Positions after the decimal point may be different. Add positions for the result position.
- Division: Positions after the decimal point may be different. Subtract REGB position from the FLAC position to get the result position.

Examples:

```
FLAC            REGB          RESULT IN FLAC

XXX.YYY   +     X.YYY            XXX.YYY
  XXX.Y   -      XX.Y              XXX.Y
XXX.YYY   x      XX.Y         XXXXX.YYYY
     XX   x        XX               XXXX
XXX.YYYY :     XX.YYY                X.Y
  XX.YY   :        XX               X.YY
```

If two numbers have to be divided and the result should have n digits after the decimal point, the FLAC has to be loaded with the number shifted to the left appropriately and zeroes filled into the lower digits. The same procedure can be used if a smaller number is to be divided by a larger one.

Examples:

FLAC		REGB	RESULT IN FLAC
XXXX.000	:	XX	XX.YYY
XXXX.000	:	XX.Y	XX.YY
XXXX.000	:	X.YY	XXX.Y
0.XXX000	:	XX.Y	0.XXXXX

It is the programmer's task to assure that no overflow conditions can occur. Before defining register length a "worst case design" has to be made concerning the greatest numbers which have to be handled. If numbers grow too large the previously described rounding routine should be used (See 6.1). If overflow occurs, no errors are reported! If division is used it must be shure, that the divisor can't be zero: The software would stay endlessly in the division loop.

```
          TITLE     INTEGER   TSS0400
*         DATE:     6.3.86
*         VERSION:  B.3
****************************************************************
*
*         INTEGER ARITHMETIC ROUTINES  FOR THE TSS400
*
****************************************************************
*
*     0       1            RRLSD    RRVZ      Y
* ------------------~~~~------------------
* |       |       |      |       |  VZ   |
* |  MSD  |       |      |  LSD  |  --   |              FLAC
* |       |       |      |       |PASSFLG|              RRXFL
* |       |       |      |       | NULFLG|
* ------------------~~~~------------------
*
* ------------------~~~~------------------
* |       |       |      |       |·      |              REGB
* |  MSD  |       |      |  LSD  |  VZ   |              RRXRB
* ------------------~~~~------------------
*
* ------------------~~~~------------------
* |       |       |      |       |  HR   |              HELP-
* |  MSD  |       |      |  LSD  |POINTER|              REG.
* ------------------~~~~------------------              RRXHR
*
* INSTRUCTION COUNT FOR THE PACKAGE:
*                   (QS = CROSS SUM     S = NUMBER OF DIGITS)
*                   (SRB= DIVIDEND LENGTH   RSLT = RESULT)
*
*                   MINIMAL             MAXIMAL
* ADDITION          4S + 15             24S + 36
* SUBTRACTION       4S + 19             24S + 40
* MULTIPLICATION    3 + 16S + 3(S x S) + QS-FLAC(4S + 10)
* DIVISION          8 + 15S + 4(S x S) + QS-RSLT(4S + 16)
*                     + (S - SRB)(3S + 11)
*
* DEFINITIONS FOR INTEGER ROUTINES
*
RRXFL     EQU       12        X OF FLAC M(RRXFL,0 - RRVZ)
RRVZ      EQU       8         Y SIGN POSITION FLAC + REGB
RRLSD     EQU       7         Y LSD FLAC, REGB AND HR
RRE1      EQU       6         Y 10E1
RRE2      EQU       5         Y 10E2
RRE3      EQU       4         Y 10E3
RRE4      EQU       3         Y 10E4
```

```
RRE5       EQU       2         Y 10E5
RRE6       EQU       1         Y 10E6
*
RRXHR      EQU       11        X HELP REG.  M(RRXHILF,0 - RRPFL)
*
*
* UNCHANGEABLE DEFINITIONS
*
RRXRB      EQU       15        X OF REGB (DAM) M(15,0 - RRVZ)
RRPFL      EQU       RRVZ      Y POINTER  M(RRXHR,RRPFL)
RRMSD      EQU       0         Y MSD FLAC, REGB AND HR
NULFLG     EQU       0         FLAGS IN FLAG REG. M(RRXFL,RRVZ)
PASSFLG    EQU       1         PASSFLAG M(RRXFL,RRVZ)
****************************************************************
*                      SUBTRACTION ROUTINE
****************************************************************
* SUBTRACTION ROUTINE WITH SIGN. FLAC = FLAC - REGB
* -------------------
* SUBTRACTION IS MADE BY ADDITION WITH NEGATED REGB.
* SIGN REMAINS CHANGED!
*
SUBTRACT   TCY       RRVZ
           XDA
           ACACC     8         CHANGE SIGN
           XDA                 FALL THROUGH TO ADDITION
****************************************************************
*                        ADDITION ROUTINE
****************************************************************
* ADDITION ROUTINE WITH SIGN. FLAC = FLAC + REGB
*-------------------------------
* REGB REMAINS UNCHANGED
*
* IF THE SIGNS OF FLAC AND REGB ARE ALIKE, NORMAL
* ADDITION CAN BE MADE.
*
ADDITION   LDX       RRXFL
ADDREGX    SAL                 SET ADDLATCH
           TCY       RRVZ
           DMEA                S-FLAC + S-REGB + SS -> AC
           ACACC     8
           BRANCH    KOMPL     SIGNS ARE DIFFERENT
*
* NORMAL ADDITION POSSIBLE /FLAC/ = /FLAC/ + /REGB/
*
NORADD     TCY       RRLSD     10E0-DIGIT
           REAC                "SPECIAL STATUS" SS = 0
ADLOP      DMEA                REGB + FLAC + SS -> AC
```

```
          TAMACS    6         AC -> FLAC, IF CARRY NOW OR
          CTMDYN              OF •DMEA• : AC + 6 -> FLAC
          BRANCH    ADLOP     ALL THROUGH?
          CCLA                LAST CARRY IN AC: SS -> AC
*
* ADDITION IS COMPLETE. CONTROL IF CORRECTION NECESSARY.
*
          TCY       RRVZ      X = RRXFL
          TBIT      PASSFLG   PASSFLAG=1?
          BRANCH    REKOMPL   YES
          BR        RETUR     FINISHED, ADD WITH SAME SIGN
*
* WHEN SIGNS ARE DIFFERENT, EVTL., FLAC MUST BE COMPLEMENTED
* ONCE MORE (NULFLG=0, NO CARRY IN AC)
*
REKOMPL   RBIT      PASSFLG   DELETE PASSFLAG
          TBIT      NULFLG    NULFLG=1?
          BRANCH    VZW
          CPAIZ
          BRANCH    NOCAR     NO CARRY FROM ADD
*
* PASSFLAG IS 1, CARRY IS 1, SIGN OF FLAG HAS TO
* BE CHANGED.
*
VZW       XMA                 Y = RRVZ
          ACACC     8
          XMA
          BR        RETUR     FINISHED
**********************************************
* SIGN OF REGB AND FLAC ARE DIFFERENT
*
KOMPL     SBIT      PASSFLG   SET PASSFLAG, Y = RRVZ
*
* COMPLEMENT OF FLAC (10ES + 1 - FLAC) => FLAC
*
NOCAR     TCY       RRLSD     LSD OF FLAC
          TCA       1         10E0 DIGIT: 10- COMPLEMENT
NILOP     ACACC     9         OTHERWISE:   9- COMPLEMENT
XMA                 FLAC DIGIT IN AC, 9(10) => M
SAMAN               9(10) - FLAC DIGIT -> AC
TAM
ACACC     6         AC > 9 ?
BRANCH    AC10      YES, CORRECTION
          CLA                 NO
          BR        ETE
AC10      TAMZA               AC + 6 IN FLAC DIGIT
          ACACC     1         CARRY FOR NEXT DIGIT
```

```
ETE       DYN
          BRANCH    NILOP
*
* FLAC IS COMPLEMENTED. TEST IF COMPLETED BECAUSE
* ROUTINE IS DOUBLE USED.
*
TCY       RRVZ      CARRY IS USED
AMAAC               AS NULFLG
TAM
TBIT      PASSFLG   IF 0 READY
BRANCH    NORADD    FOR ADDITION
RETUR     RBIT      NULFLG
          RETN
*
* END OF ADDITION ROUTINE
****************************************************************
*                    MULTIPLICATION ROUTINE
****************************************************************
* MULTIPLICATION ROUTINE WITH SIGN
*--------------------------------------
* FLAC = FLAC * REGB. REGB REMAINS UNCHANGED
*
MULTIPLY  LDX       RRXFL     SIGN OF RESULT --> FLAC
          TCY       RRVZ
          SAL                 SET "ADD LATCH"
          REAC                SS = 0     Y = RRVZ
          DMEA                VZ-FLAC + VZ-REGB + SS -> AC
          TAMDYN              VZ -> VZ OF FLAC. Y = RRLSD
*
FLHR      LDX       RRXFL     FLAC -> HELP REGISTER
          CLA
          XMA                 CLEAR FLAC
          LDX       RRXHR
          TAMDYN
          BRANCH    FLHR
*
          TCY       RRPFL     X = RRXHR
          TCMIY     RRMSD     SET POINTER TO MSD
*
* REGB IS ADDED TO THE FLAC AS OFTEN AS IS GIVEN BY
* THE NUMBER IN THE HR-DIGIT. START WITH MSD.
*
MULLOP    LDX       RRXHR
          TCY       RRPFL     PRESENT HR-DIGIT
          TMY                 ADDRESS + DECREMENT
          DMAN                ALREADY 0?
          BRANCH    FADD      NO, ADD. ONCE MORE
```

```
*
* THE PRESENT HR-DIGIT IS 0. MULTIPLICATION IS
* FINISHED IF POINTER POINTS TO 10E0 HR, OTHERWISE FLAC
* IS MULTIPLIED BY 10 (SHIFT LEFT, 1 POSITION).
*
          YNEC      RRLSD     10E0 DIGIT HR ?
          BRANCH    FLSH      NO: FLAC x 10
*
* 10E0 IS PROCESSED. RESULT WITHIN FLAC, FINISHED.
*
          RETN
*
* REGB IS ADDED TO THE FLAC.
*
FADD      TAMZA               HR-DIGIT - 1 -> HR
          TCY       RRLSD     ADDITION
          LDX       RRXFL
          REAC                SS=0 (SAL SETS SS !)
ADLOPM    DMEA                FLAC + REGB + SS -> AC
          TAMACS    6         /FLAC/ = /FLAC/ + /REGB/
          CTMDYN
          BRANCH    ADLOPM
          BR        MULLOP    ADDITION FINISHED
*---------
* SHIFT FLAC 1 DIGIT TO THE LEFT (X 10)
*
FLSH      TCY       RRPFL     X = RRXHR
          IMAC                POINTER FOR HR
          TAMZA               TO NEXT LOWER DIGIT
*
          LDX       RRXFL
          TCY       RRLSD     AC = 0
FLSH1     XMA
          DYN
          BRANCH    FLSH1
          BR        MULLOP
*
* END OF MULTIPLICATION
****************************************************************
*                      DIVISION ROUTINE
****************************************************************
* WITH SIGN. FLAC = FLAC/REGB. REMAINDER IN HR
*
DIVISION  LDX       RRXFL     BUILD SIGN OF RESULT
          TCY       RRVZ
          REAC                RESET SS
          DMEA
          TAMDYN                Y = RRLSD
*
HRRLOP    LDX       RRXFL       FLAC -> HR
          CLA                   FLAC <-- 0000
          XMA
          LDX       RRXHR
          TAMDYN
          BRANCH    HRRLOP
*
          TCY       RRPFL       X = RRXHR
          TCMIY     RRLSD       SET POTCTR TO LSD
*
* SHIFT CONTENT OF REGB TO THE LEFT, UNTIL MSD IS UNLIKE 0.
*
INLOP     LDX       RRXRB
          TCY       RRMSD
          MNEZ
          BRANCH    PDIVLOP     UNLIKE 0: FINISHED
*
* MSD IS STILL 0. REGB IS SHIFTED 1 DIGIT TO THE LEFT.
*
          TCY       RRLSD
          CLA                   0 -> LSD
REGSHL    XMA
          DYN
          BRANCH    REGSHL
*
          LDX       RRXHR       POINTER TO NEXT HIGHER DIGIT.
          TCY       RRPFL
          DMAN
          TAMZA
          BR        INLOP
*
PDIVLOP   BL        DIVLOP
          PAGE
RB:10     TCY       RRMSD       DIVIDE REGB BY 10
          CLA
RSHLOP    XDA
          IYC
          YNEC      RRVZ
          BRANCH    RSHLOP
*
* CHECK IF HR > REGB
*
DIVLOP    TCY       RRMSD
          LDX       RRXHR
COMLOP    TDA
```

```
          ALEM                  REGB-DIGIT IN AC
          BRANCH     FLLE       HR-DIGIT >= REGB-DIGIT
          BR         DEPO       HR < REGB
FLLE      MNEA
          BRANCH     ADDO       HR > REGB
          IYC                   DIGITS EQUAL, NEXT
          YNEC       RRVZ
          BRANCH     COMLOP     HR-DIGIT = REGB-DIGIT
*
* HR >= REGB-DIGIT POINTS TOWARDS POWER POINTER
* REGB IS SUBTRACTED FROM HR
*
ADDO      TCY        RRLSD      X = RRXHR
          SEAC                  SS = 1
SUBLP     NDMEA                 -REGB + FLAC + SS -> AC
          TAMACS     10
          CTMDYN
          BRANCH     SUBLP
*
          TCY        RRPFL      X = RRXHR
          TMY
          LDX        RRXFL
          IMAC                  INCR. DIGIT IN FLAC
          TAMZA
          BR         DIVLOP
*
* HR < REGB: POTCTR IS INCREMENTED. IF IT THEN
* POINTS TOWARDS 10E-1 (RRVZ), DIVISION IS FINISHED.
*
DEPO      TCY        RRPFL      X = RRXHR
          IMAC
          TAM
          ALEC       RRLSD
          BRANCH     RB:10
*
* POTCTR = RRVZ: DIVISION FINISHED.
* RESULT IS IN FLAC, REMAINDER IN HELP REGISTER.
*
          RETN                  FINISHED
*
* END OF DIVISION ROUTINE       END OF INTEGER ROUTINES
*--------------------------------------------------------------
```

6.6 Square Root Routine

The square root is often needed in computations. The following subroutine uses the NEWTONIAN approximation for this problem. 10 iterations are used.

```
* SQUARE ROOT OF VALUE  A IN FLAC: X = SQROOT(A)
* Xn+1 = 1/2(Xn + A/Xn) NEWTONIAN APPROXIMATION
* BEFORE CALL:      A IN FLAC  XXXXX OR  XXXX.XX OR XXX.XXXX
* AFTER RETURN:     X IN FLAC  XXX.Y OR   XXX.YY      XX.YYY
*
SQROOT    CALLL     FLACZWSP  A     -> ZWSP            XXXX
          CALL      FLTORB    A/10 -> REGB              XXX.X
          TCY       LOOPCNT   X = RRXFL
          TCA       10        SET LOOP COUNT = 10
L$004     TAM
          CALLL     CLRFLAC   +0 -> FLAC
          CALLL     ZWFL100   A x 100 -> FLAC         XXXX.00
          CALLL     DIVISION  A/Xn     -> FLAC         XXX.Y
          CALLL     ADDITION  A/Xn + Xn -> FLAC        XXX.Y
          CALLL     CLRREGB
          TCY       RRLSD
          TCMIY     2         2 -> REGB                  X
          CALLL     DIVISION  Xn+1 = 1/2(A/Xn + Xn)  XXX.Y
          CALL      FLTORB    Xn+1 -> REGB             XXX.Y
          TCY       LOOPCNT   X = RRXFL
          DMAN                LOOP COUNT  - 1
          BRANCH    L$004
          RETN
*---------
ZWFL100   TCY       RRLSD     ZWSP x 100 -> FLAC
L$005     LDX       XZWSP     M(XZWSP,0-RRVZ) -> M(RRXFL,0-RRVZ)
          TMA
          ACYY      14
          LDX       RRXFL
          TAMIYC
          YNEC      1         Y MSD-1
          BRANCH    L$005     XXXX.00
          RETN
*
FLTORB    LDX       RRXFL     COPY FLAC TO REGB
          TCY       RRVZ
L$006     TMA
          XDA                 M(RRXFL,0-RRVZ) -> M(15,0-RRVZ)
          DYN
          BRANCH    L$006
          RETN
*
FLACZWSP  TCY       RRVZ      COPY FLAC TO INTERM. BUFFER
L$009     LDX       RRXFL
          TMA
          LDX       XZWSP     M(RRXFL,0-RRVZ)  -> M(15,0-RRVZ)
          TAMDYN
          BRANCH    L$009
          RETN
```

6.7 ADC Measurement Routines

To achieve a reliable result A/D conversions should add at least two measurements: one with noninverted and one with inverted comparator inputs. This compensates the offset error of the comparator. Adding more than two measurements increases the accuracy of the measurements. After each conversion the range bit should be controlled.

Three measurement routines are shown:
- Two measurements
- More than two measurements with fixed count
- More than two measurements with count in accumulator

The routines use all the same subroutines and are optimized concerning ROM space. Implemented delays are for the highest MOS oscillator frequency (950 kHz). The SV_{DD} is switched on as soon as possible to allow the resistor network to settle.

If the highest possible speed is necessary, the measurement loop is to be coded straight forward. This means, the successive approximation part of the following examples would look like:

```
MEASLOOP  LDX       15
          TCY       13        ADC HW-BUFFER  M(15,13-15)
          TCMIY     8         LOADED WITH MIDTH OF RANGE    -
          TCMIY     0         (800)                         |
          TCMIY     0                                       |
          TCY       13        START WITH MSB OF MSD       7 INSTR.
          CALL      DELAY3                                  V
          TCTM      3                                       -
          CALL      DELAY4                                5 INSTR.
          TCTM      2                                       -
          CALL      DELAY3                               4 INSTR.
          TCTM      1                                      -
          CALL      DELAY2                               3 INSTR.
          TCTM      0                                      -
          IYC                 M(15,14)
          SBIT      3                                      -
          MNEA                DELAY                      2 INSTR.
          TCTM      3                                      -
          TCTM      2         1 INSTRUCTION DELAY
          TCTM      1         UNTIL LSB
          TCTM      0
          IYC                 M(15,15)
          SBIT      3
          TCTM      3
          TCTM      2
          TCTM      1
          TCTM      0         LSB: CONVERSION COMPLETE
```

6.7.1 ADC Measurement Routines with two Measurements

```
* MEASUREMENT ROUTINE: 2 MEASUREMENTS ARE ADDED, ONE WITH
* INVERTED AND ONE WITH NONINVERTED COMPARATOR INPUTS.
* M(15,12) IS USED FOR COUNTING
* RETURN:     AC = 0: MEASUREMENT OK, RESULT M(XA/D,12-15)
*             AC # 0: MEASUREMENT NOT OK, RESULT UNDEFINED
*
* CALL:   CALLL  MEASINIT     1. MEASUREMENT
*         CALLL  MEASURE      FOR ADDITIONAL MEASUREMENTS
*
MEASINIT  SCSV                SWITCH SVDD ON, COMP. NONINV.
          LDX      XA/D       0 -> A/D BUFFER M(XA/D,12-15)
          TCY      12         (BUFFER FOR ADDING)
          TCMIY    0
          TCMIY    0
          TCMIY    0
          TCMIY    0
*
MEASURE   TCY      DL13       CURRENT SOURCE ON (DELETE LINES
          SETR                IF VOLTAGE SOURCE NEEDED)
*
LDX       15       1. MEASUREMENT: COMPARATOR NONINV.
          TCY      12         LOAD COUNTER WITH N-1 (1)
          TCMIY    1
*
MEASLOOP  LDX      15
```

```
          TCY       13        ADC HW-BUFFER  M(15,13-15)
          TCMIY     8         LOADED WITH MIDTH OF RANGE     -
          TCMIY     0         (800)                          |
          TCMIY     0                                        |
          TCY       13        START WITH MSB OF MSD        7 INSTR.
SUKZLP    SBIT      3         SUCCESSIVE LOOP M(15,13-15)    |
          CALL      DELAY2                                   V
          TCTM      3                                        -
          CALL      DELAY4                                 5 INSTR.
          TCTM      2                                        -
          CALL      DELAY3                                 4 INSTR.
          TCTM      1                                        -
          CALL      DELAY2                                 3 INSTR.
          TCTM      0                                        -
          IYC
          YNEC      0
          BRANCH    SUKZLP
*
          TCY       12        RANGE CONTROL:
          TRTM      3         RESULT >FFF ODER >000 ?
          TBIT      3         M(15,12,3): 0 = OK
          BRANCH    MESERROR              1 = FEHLMESSUNG
          DMAN                COUNT -1
          BRANCH    M2
*
* BOTH MEASUREMENTS ARE OK: ADD RESULT WITH INVERTED
* COMPARATOR INPUTS TO PREVIOUS RESULT.
*
          RCSV                SWITCH SVDD OFF
ADDA/D    TCY       15        ADD A/D VALUE M(15,13-15)
          LDX       XA/D      TO A/D BUFFER M(XA/D,12-15)
          SAL
          REAC
ADL       DMEA
          TAMACS    0
          CTMDYN
          YNEC      12
          BRANCH    ADL
          CCLA                SPECIAL STATUS -> A
          AMAAC               ADD MSD
          TAMZA               MSD -> M(12,12)
          RETN                AC = 0: ALL OK
*---------
* MEASUREMENT WITH NONINVERTING COMPARATOR INPUTS IS OK:
* STORE A/D VALUE.
* START MEASUREMENT WITH INVERTED COMPARATOR INPUTS NOW
*
```

```
M2        TAM
          CALL      ADDA/D    A/D VALUE -> A/D BUFFER
          RCI                 INV. COMP. INPUTS
          BR        MEASLOOP  ONCE MORE
*---------
* MEASUREMENT ERROR: AC <- 15
*
MESERROR  RCSV                SWITCH SVDD OFF
          TCA       15        ERROR INDICATION -> A
          RETN
*---------
* DELAY ROUTINES: THE CALL INSTRUCTION IS COUNTED TOO
*
DELAY4    MNEA                4 INSTRUCTIONS DELAY
DELAY3    MNEA                3 INSTRUCTIONS DELAY
DELAY2    RETN                2 INSTRUCTIONS DELAY
```

6.7.2 ADC Measurement Routine with "EQU" defined Number of Conversions

The next A/D conversion uses a fixed number of measurements defined by an "EQU" directive. The maximum number of added conversions is 4 · 2. If more added conversions are needed, the entry "MEASURE" can be used once more for additional 4 · 2 measurements (then the four-digit A/D buffer is full).

```
* ANALOG DIGITAL CONVERSION ROUTINE
* NUMBER OF CONVERSIONS IS DEFINED BY "EQU" DIRECTIVE
* M(15,12) IS USED FOR COUNTING
* RETURN:    AC = 0: MEASUREMENT OK, RESULT IN M(XA/D,12-15)
*            AC # 0: MEASUREMENT NOT OK, RESULT UNDEFINED
*
* CALL:      CALLL  MEASINIT  1st N MEASUREMENTS
*            CALLL  MEASURE   ADDITIONAL N MEASUREMENTS
*
* ENTRY FOR 1st N MEASUREMENTS
*
N-1       EQU       7         EXAMPLE: ADD UP 8 MEASUREMENTS
*
MEASINIT  SCSV                SWITCH SVDD ON, COMP. NONINV.
          LDX       XA/D      0 -> A/D BUFFER M(XA/D,12-15)
          TCY       12        (BUFFER FOR ADDING UP)
          TCMIY     0
          TCMIY     0
          TCMIY     0
          TCMIY     0
*
* ENTRY FOR ADDITIONAL N MEASUREMENTS
*
MEASURE   TCY       DL13      CURRENT SOURCE ON
          SETR
*
          LDX       15        1st MEASUREMENT: COMP. NONINV.
          TCY       12        COUNTER FOR N MEASUREMENTS
          TCMIY     N-1       LOAD COUNTER WITH N-1 (Nmax = 8)
*
MEASLOOP  LDX       15
          TCY       13        LOAD ADC HW-BUFFER  M(15,13-15)
          TCMIY     8         WITH MIDTH OF ADC RANGE
          TCMIY     0
          TCMIY     0
          TCY       13        START WITH MSB OF MSD
SUKZLP    SBIT      3         SUCCESSIVE LOOP M(15,13-15)
          CALL      DELAY2
          TCTM      3
          CALL      DELAY4
          TCTM      2
          CALL      DELAY3
          TCTM      1
          CALL      DELAY2
          TCTM      0
          IYC
          YNEC      0
          BRANCH    SUKZLP
*
          TCY       12        RANGE CONTROL:
          TRTM      3         RESULT >FFF ODER >000 ?
          TBIT      3         M(15,12,3): 0 = OK
          BRANCH    MESERROR              1 = ERROR
          DMAN                DECREMENT COUNT
          BRANCH    M2
*
* ALL MEASUREMENTS OK: ADD RESULT TO A/D BUFFER
*
          RCSV                SWITCH SVDD OFF
ADDA/D    TCY       15        ADD A/D VALUE M(15,13-15)
          LDX       XA/D      TO A/D BUFFER M(XA/D,12-15)
          SAL
          REAC
ADL       DMEA
```

```
          TAMACS    0
          CTMDYN
          YNEC      12
          BRANCH    ADL
          CCLA                SPECIAL STATUS -> AC
          AMAAC               ADD MSD
          TAMZA               MSD -> M(12,12)
          RETN                AC = 0: ALL OK
*---------
* MEASUREMENT OK. SWITCH COMPARATOR INPUT:
* EVEN MEASUREMENTS WITH INVERTED COMPARATOR INPUTS,
* ODD MEASUREMENTS WITH NONINVERTED COMPARATOR INPUTS.
*
M2        TAM                 WRITE BACK DECREMENTED COUNT
          SCSV                NON INVERTED COMP. INPUTS
          TBIT      0         BIT 2 0 CONTAINS COMP. INFO
          BRANCH    L$810     ODD MEASUREMENT: NONINV.
          RCI                 EVEN MEASUREMENT: INV.
L$810     CALL      ADDA/D    A/D VALUE -> A/D BUFFER
          BR        MEASLOOP  NEXT MEASUREMENT
*----------
* ERRONEOUS MEASUREMENT: AC <-- 15
*
MESERROR  RCSV                SWITCH OFF SVDD
          TCA       15        ERROR INDICATION -> A
          RETN
```

6.7.3 ADC Measurement Routine with Number of Conversions in Accumulator

The next A/D conversion uses a number of measurements which is contained in the accumulator. The maximum number of added conversions is 8. If more added conversions are needed, the entry "MEASURE" can be used once more for additional 8 measurements (then the four-digit A/D buffer is full). Starting at the label MEASLOOP the same software is used as with the last example.

```
* ANALOG DIGITAL CONVERSION ROUTINE
* NUMBER OF CONVERSIONS N IS CONTAINED IN ACCUMULATOR AS N-1
* M(15,12) IS USED FOR COUNTING
* RETURN:     AC = 0: MEASUREMENT OK, RESULT IN M(XA/D,12-15)
*             AC # 0: MEASUREMENT NOT OK, RESULT UNDEFINED
*
```

```
* CALL:        TCA    N-1        CALL FOR N MEASUREMENTS
*              CALLL  MEASINIT   1st N MEASUREMENTS
               TCA    N-1
*              CALLL  MEASURE    ADDITIONAL N MEASUREMENTS
*
* ENTRY FOR 1st N MEASUREMENTS
*
MEASINIT  SCSV                   SWITCH SVDD ON, COMP. NONINV.
          LDX       XA/D         0 -> A/D BUFFER M(XA/D,12-15)
          TCY       12           (BUFFER FOR ADDING UP)
          TCMIY     0
          TCMIY     0
          TCMIY     0
          TCMIY     0
*
* ENTRY FOR ADDITIONAL N MEASUREMENTS
*
MEASURE   TCY       DL13         CURRENT SOURCE ON
          SETR
*
          LDX       15           1st MEASUREMENT: COMP. NONINV.
          TCY       12           COUNTER FOR N MEASUREMENTS
          TAM                    LOAD COUNTER WITH N-1 FROM ACCU
*
MEASLOOP  LDX       15
          TCY       13           LOAD ADC HW-BUFFER  M(15,13-15)
          ...                    PROGRAM EXACTLY LIKE SHOWN ABOVE
          ...                    SEE EXAMPLE 6.7.2
```

6.8 Binary to BCD Conversion

The hexadecimal value in M(XA/D, 12-15), normally the output value of an A/D conversion, is converted to BCD format by this subroutine. The result is stored in the FLAC. The memory nibble M(XA/D,YPOI) is used as a pointer to the hexadecimal number. The subroutines of the Integer Math. Package are used for the conversion. If hex numbers are greater than >FFFF, the pointer M(XA/D, YPOI) must be loaded initially with the Y address of the MSD.

```
* CONVERSION OF A HEXADECIMAL VALUE INTO DECIMAL FORMAT
*
*        INPUT: HEX VALUE (0 - >FFFF)  IN M(XA/D,12-15)
* AFTER RETURN: BCD VALUE (0 - 65535)  IN         FLAC
```

```
*
HEXDEC    CALL      CLRFLAC    +0 -> FLAC
          LDX       XA/D
          TCY       YPOI       SET POINTER FOR ACTUAL HEX DIGIT
          TCMIY     12         TO MSD OF THE HEX NUMBER
*
L$045     LDX       XA/D       POINTER -> Y
          TCY       YPOI
          TMY
          TMA                  ACTUAL HEX DIGIT -> AC
          CALL      CLRREGB    +0 -> REGB
          TCY       RRLSD      ADD ACTUAL HEX DIGIT TO FLAC
          TAM
          ALEC      9          CORRECT IT IF GREATER THAN 9
          BRANCH    L$046
          ACACC     6          >A TO >F -> 10 TO 15
          TAMDYN
          TCMIY     1
L$046     CALLL     ADDITION   FLAC + REGB -> FLAC
          LDX       XA/D       TO NEXT LOWER HEX DIGIT
          TCY       YPOI
          IMAC                 POINTER := POINTER + 1
          BRANCH    L$047      IF RESULT IS 0: CONV. DONE
          TAM                  POI +1 -> POI
          CALL      CLRREGB
          TCY       RRE1       16 -> REGB
          TCMIY     1
          TCMIY     6
          CALLL     MULTIPL    FLAC x 16 -> FLAC
          BR        L$045
L$047     RETN                 RESULT IN FLAC
```

6.9 Keyboard Scan Routines

If more than 2 keys are used in an application, the keys are normally arranged in a matrix to save outputs and inputs. With n outputs and m inputs, m · n keys, programming diodes and switches can be scanned.

Three different hardware configurations are possible:
- If only one key can be activated at any time, no diodes are necessary for decoupling.
- If multiple key functions are not allowed (but cannot be excluded) diodes are necessary only at each output.

- If multiple key functions are used, each key needs a decoupling diode in series with the contact.

Two software solutions for keyboard scanning are possible:
- If the wakeup frequency is 8 Hz or higher, polling is possible after wakeup. Debouncing can be done by multiple wakeups, which results in current saving compared to delay routines.
- If the wakeup frequency is 1 Hz, the R-outputs which control keys, have to be set during Done Mode to allow wakeup by key activation.

6.9.1 Keyboard Routine with fast Wakeup [8 to 128 Hz]

The example uses one RAM nibble for each R-output and a common status nibble for all keys. The status nibble is shift left one position by each wakeup and the LSB is set if a keyboard change occured, compared to the last wakeup. If no change occured the LSB is reset.

The debouncing time is one wakeup period.

The following information can be extracted out of the status nibble:

Bit 1	Bit 0	Function
0	0	no change compared with last wakeup
0	1	change occured, but not yet debounced
1	0	change occured, debounced
1	1	change ongoing

Figure 20: Information in Keyboard Status Bits

For wakeup frequencies higher than 8 Hz, Bit 2 and Bit 3 of the status nibble can also be used:

Debouncing is reached if several LSBs are all 0 and the next bit is 1.

The K and R lines of the RAM nibbles 0-2 show the current state of the keyboard. Each activated key, switch or programming diode is represented by a set bit. All other bits are reset:

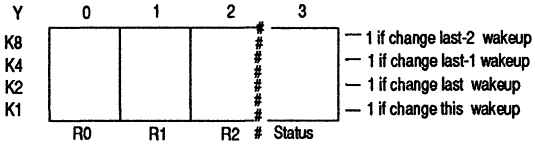

Y
0
1
2
3
K8
K4
K2
K1
R0
R1
R2
Status
— 1 if change last-2 wakeup
— 1 if change last-1 wakeup
— 1 if change last wakeup
— 1 if change this wakeup

Figure 21: RAM Organization Keyboard Routines

The routine is designed for applications where DL12 is always set.This means wakeup frequencies from 8 Hz to 128 Hz. The switches, diodes and keys are placed in a 3 · 4 matrix: (R0, R1, R2) and (K1, K2, K4, K8). The high wakeup frequency allows normal scanning of the keyboard without using the wakeup feature of the K-Port.

```
* KEYBOARD ROUTINE WITH FAST WAKEUP
*
XORG      EQU       1         X BANK PROGRAM ORGANIZATION
R0        EQU       0         Y FOR R0
R1        EQU       1         Y FOR R1
R2        EQU       2         Y FOR R2
KEYSTAT   EQU       3         Y FOR KEYBOARD STATUS
*
          ORG       1927      WAKEUP POINT PAGE 15 PC 7F
*
WAKEUP    RETN                RESET INTERNAL LATCHES
          CALL      KEYBOARD  SCAN KEYBOARD
*
TCA       12        STATUS NIBBLE IS ADDRESSED
ORMA                STATUS .OR. MASK 1100 -> A
ALEC      13        1100 OR 1101 ?
BRANCH NOCHANGE     NO CHANGE OR NOT DEBOUNCED
*
ALEC      14        1110 ?
BRANCH    CHANGE    DEBOUNCED CHANGE: PROCESS IT
*
CHANGEON  ...                 1111: CHANGE ONGOING
          ...
*
NOCHANGE  ...                 NO KEYBOARD CHANGE:
          ...                 PROCESS TIMER INFO
*
CHANGE    ...                 PROCESS KEYBOARD CHANGE
```

```
          DONE                      AND SLEEP AGAIN
*---------
* KEYBOARD ROUTINE
*
KEYBOARD  LDX        XORG           M(XORG,0-3) USED FOR KEYBOARD
          TCY        R0             START WITH R0 KEYS
L$500     SETR                      SET R OUTPUT
          TMA                       INFO OF LAST WAKEUP -> A
          TKM                       NEW INFO -> MEMORY
          RSTR                      RESET R OUTPUT
          MNEA                      CHANGE COMPARED TO LAST WAKEUP ?
          BRANCH     L$501
          IYC                       NO, PREPARE FOR NEXT R OUTPUT
          YNEC       KEYSTAT        ALL R OUTPUTS THROUGH ?
          BRANCH     L$500
          TCA        0              YES, NO CHANGE IN KEYBOARD
          BRANCH     L$502          RESET BIT 0 OF STATUS
*
L$501     TCA        1              CHANGE IN KEYBOARD, SET BIT 0
L$502     TCY        KEYSTAT
          AMAAC                     SHIFT LEFT PREVIOUS INFO
          AMAAC                     NEW STATUS TO BIT 0
          TAM                       RETURN WITH ADDRESSED STATUS
          RETN                      NIBBLE M(XORG,KEYSTAT)
```

6.9.2 Keyboard Routine with 1 Hz Wakeup

The example uses the same RAM locations and status information as the "fast version" described before. The R-outputs are set to allow fast response to a key activation. After recognition of a LO-HI transition at any K-input, the Digit Latch DL12 is set to operate at a higher wakeup frequency for debouncing. The keyboard scanning is made as described in the "fast version" before. When all keys are released as debounced, DL12 is reset to 1 Hz wakeup frequency and the R-outputs are set again to allow wakeup by LO-HI transitions.

Note: R-outputs which control switches and diodes must not be set during Done Mode: wakeup initiated by LO-HI transitions is only possible, if all other K-inputs are low.

```
* KEYBOARD ROUTINE WITH 1 HZ WAKEUP
*
XORG      EQU       1         X BANK PROGRAM ORGANIZATION
R0        EQU       0         Y FOR R0
R1        EQU       1         Y FOR R1
R2        EQU       2         Y FOR R2
KEYSTAT   EQU       3         Y FOR KEYBOARD STATUS
DL12      EQU       12        0: 1 HZ WAKEUP  1: 8 TO 128 HZ
*
          ORG       1920      START AFTER POWER UP
POWERUP   RETN
          CALL      INIT      CLEAR RAM
          ...                 PROCEED WITH INITIALIZATION
          CALL      SETR012   SET R0, R1 AND R2 TO ALLOW
          DONE                WAKEUP BY KEY ACTIVATION
*
          ORG       1927      WAKEUP POINT PAGE 15 PC 7F
WAKEUP    RETN                RESET INTERNAL LATCHES
          TCY       R2        RESET R2 AND R1 TO ALLOW SCAN
          RSTR
          TCY       R1
          RSTR
          CALL      KEYBOARD  SCAN KEYBOARD FIRST
*
          TCA       12        STATUS NIBBLE IS ADDRESSED
          ORMA                STATUS .OR. MASK 1100 -> A
          ALEC      12        1100 ?
          BRANCH    NOCHANGE  KEYBOARD INACTIVE
*
          ALEC      13        1101 ?
          BRANCH    CHANGEON  NOT DEBOUNCED CHANGE
*
ALEC      14        1110 ?
BRANCH    CHANGE    DEBOUNCED CHANGE: PROCESS IT
*
CHANGEON  TCY       DL12      1101 OR 1111: CHANGE ONGOING
          SETR                FAST WAKEUP FOR DEBOUNCING
          BR        MAINLOOP
*
NOCHANGE  TCY       R0        NO KEYBOARD CHANGE: 1 HZ WAKEUP
L$505     MNEZ                IF ALL KEYS ARE RELEASED
          BRANCH    MAINLOOP  PROCESS KEY INFO
          IYC
          YNEC      KEYSTAT
          BRANCH    L$505
*
          TCY       DL12      NO KEY ACTIVATED
          RSTR                  WAKEUP BACK TO 1 HZ
          CALL      SETR012     SET R-OUTPUTS
          BR        MAINLOOP
*--------
CHANGE    ...                   PROCESS KEYBOARD CHANGE
          DONE                  AND SLEEP AGAIN
*---------
* SUBROUTINE SETS R0, R1 AND R2 TO ALLOW WAKEUP
*
SETR012   TCY       R2
L$506     SETR
          DYN
          BRANCH    L$506
          RETN
```

6.10 Clock Routine

The subroutine maintains the time in the european format:

00.00.00 to 23.59.59

The subroutine is entered with the difference in seconds to the last call in the accumulator. This ensures that no second is lost up to 15 seconds difference.

```
XCLK      EQU       3          X BANK OF CLOCK
YSEC0     EQU       5          Y SECONDS 10E0
YHRS0     EQU       1          Y HOURS   10E0
YHRS1     EQU       0          Y HOURS   10E1
YHRS1-1   EQU       15         Y OUTSIDE CLOCK BUFFER
*
CLOCK     LDX       XCLK        DIFFERENCE IN ACCUMULATOR
          TCY       YSEC0       SEC 10E0
          SAL                   SET ADD LATCH FOR ADDITION
CLKLP     AMAAC
          TAMACS    6           SECONDS, MINUTES, HOURS 10E0
          CTMDYN                CORRECT FOR 0 - 9
          CCLA                  SS (CARRY) -> A
          AMAAC
          TAMACS    10          SECONDS, MINUTES, HOURS 10E1
          CTMDYN                CORRECT FOR 0 - 6
          CCLA
          YNEC      YHRS1-1     HOURS 10E1 YET DONE ?
          BRANCH    CLKLP       NO
    *
```

```
          TCY       YHRS0     TIME = 24.XX ?
          TMA
          ALEC      3
          BRANCH    CLKRET
          TCY       YHRS1
          TMA
          ALEC      1
          BRANCH    CLKRET
          TCMIY     0         YES, CORRECT TO 00.XX
          TCMIY     0
CLKRET    RETN
```

6.11 Temperature Computation for Sensors

6.11.1 Temperature Computation for PT500 Sensors

```
* SUBROUTINE: COMPUTATION OF SENSOR TEMPERATURE FROM A/DC VALUE
*
* INPUT:        ADC VALUE    (>A - >9FEC)     IN M(XA/D,12-15)
* AFTER RETURN: TEMPERATURE T (+-XXX.YYYY C)   IN FLAC
*
* VALUES FOR COMPUTATION:
*
* RV       = 487   OHM            Rext  CURRENT SOURCE RESISTOR
* R25      = 500   OHM                  SENSOR RESISTANCE @ 0 C
*
* ADC VALUE 0002 EQUALS  0.231314 x SVDD   LOWEST  A/D VALUE
* ADC VALUE 1FFC EQUALS  0.407511 x SVDD   HIGHEST A/D VALUE
*
* FROM THE ABOVE:
* ADC VALUE N = Vin/SVDD x 46459.36 - 10744.7     ADC EQUATION
*
* USED VALUES FOR COMPUTATION:
*
*        TEMP [C]        RF              ADC VALUE
*      -------------------------------------------
*          0            500               3572.718
*        (35)          (568.05)          (11370.73)  CONTROL ONLY
*         70            635.35           19082.81
*        140            767.9            34272.03
*        200            879.2            47026.17
*
*        T = (((A x M + B) x M + C) x M) + D
*
```

```
* WHERE   A = + 7.44762 E -15     CUBIC TERM
*         B = + 2.68166 E -9      QUADRATIC TERM
*         C = + 4.44912 E -3      LINEAR TERM
*         D = - 1.59343 E +1      CONSTANT TERM
*
*         M = + 10.....40940      A/D VALUE FROM 10 CONVERSIONS
*
COMPUTE   CALLL     HEXDEC    HEX A/DC VALUE -> BCD FLAC
          TCY       RRE4
L$200     LDX       RRXFL     TRANSFER M FROM FLAC
          TMA                 TO M(XA/D,10-15)
          LDX       XA/D
          TAMIYC
          YNEC      15
          BR        L$200     M IS IN FLAC                XXXXX
*
          CALLL     CLRREGB   CONST. A TO REGB
          TCY       RRE2      7.44762 E -15
          TCMIY     7         E -15
          TCMIY     4                    0.00|000000000000YYY
          TCMIY     4
          CALLL     MULTIPL   A x M      0.00|000000000YYYYYY
*
          CALLL     CLRREGB   CONST. B TO REGB
          TCY       RRE8      2.68166 E -9
          TCMIY     2         E -9
          TCMIY     6                    0.00|000000YYYY00000
          TCMIY     8
          TCMIY     2
          CALLL     ADDITION  A x M  + B 0.00|000000YYYYYYYYY
*
          CALLL     CLRREGB   M TO REGB
          CALLL     LOADM                                 XXXXX
          CALLL     MULTIPL   Mx(AxM+B)  0.00|000YYYYYYYYYYYY
*
          CALLL     DIV10E4   SHIFT RIGHT 6 TIMES
          CALLL     DIV10E2                     0.00000YYYYYY
          CALLL     CLRREGB
          TCY       RRE8      CONST. C TO REGB   4.44912 E -3
          TCMIY     4         E -3
          TCMIY     4                           0.00YYYYYY000
          TCMIY     4
          TCMIY     9
          TCMIY     1
          TCMIY     2
          CALLL     ADDITION  Mx(AxM+B)+C       0.00YYYYYYYYY
          CALLL     CLRREGB   M TO REGB
          CALLL     LOADM                                       MMMMM
          CALLL     MULTIPL   Mx(Mx(AxM+B)+C)  XX.YYYYYYYYYYY
          CALLL     CLRREGB   CONST. D TO REGB  -1.593 E +1
          TCY       RRE12
          TCMIY     1         E +1              X.YYY00000000
          TCMIY     5
          TCMIY     9
          TCMIY     3
          TCY       15
          TCMIY     8         INSERT - SIGN
          CALLL     ADDITION  Mx(Mx(MxA)+B)+C)+D
          CALLL     DIV10E3
          CALLL     DIV10E4   SHIFT 7 RIGHT
          RETN                                              XXX.YYYY
```

6.11.2 Temperature Computation for TSP202 Sensors

```
* SUBROUTINE COMPUTES TEMPERATURE FROM A/DC VALUE
*
* INPUT:        ADC VALUE    (>2 - >1FFC)  IN M(XA/D,12-15)
* AFTER RETURN: TEMPERATURE T (+-XXX.YY C)  IN FLAC
*
* BASE OF COMPUTATIONS:
*
* RV        = 5.11 KOHM                         SERIES RESISTOR
* R25       = 2.00 KOHM                         SENSOR RESISTOR @25C
* R85/R25   = 1.54                              SENSOR INCLINATION
* ADC VALUE 0002 EQUALS  0.231271 x SVDD        LOWER  A/D VALUE
* ADC VALUE 1FFC EQUALS  0.407511 x SVDD        UPPER  A/D VALUE
* FROM THE ABOVE:
* ADC VALUE = Vin/SVDD x 46448.18355 - 10740.13644 A/DC EQUATION
* T = ADC VALUE x 0.013629859 - 6.695646691     EXACTLY
* T = ADC VALUE x 0.01363     - 6.696           USED
*
*  CALL:  ADC VALUE   (0 - >FFFF)   IN M(XA/D,12-15)
* RESULT: TEMPERATURE (XXX.YY)      IN FLAC
*
TEMPCOMP CALLL     HEXBCD    COMPUTE BCD VALUE OF (XA/D,12-15)
         CALLL     CLRREGB   BCD VALUE IN FLAC       XXXXX
         TCY       RRE3
         TCMIY     1         0.01363 -> REGB              0.0YYYY
         TCMIY     3
         TCMIY     6
         TCMIY     3
         CALLL     MULTIPL   ADC VALUE x 0.01363        XXX.YYYYY
```

```
        CALLL     ROUNDFL   ROUND FLAC                   XXX.YYYY
        CALLL     ROUNDFL                                XXX.YYY
        CALLL     CLRREGB
        TCY       RRE3
        TCMIY     6         6.696-> REGB                   X.YYY
        TCMIY     6
        TCMIY     9
        TCMIY     6
        CALLL     SUBTRAKT  ADC VALUE x 0.01363 - 6.696  X.YYY
        CALLL     ROUNDFL                              +-XXX.YY
        RETN
```

6.12 Battery Check Routines

6.12.1 Battery Check Routine I

The 1st subroutine is used during system calibration with $V_{DD} = V_{DDmin}$. The output of the measurement routine is stored in M(10,12-15) for later references by the 2nd subroutine.

```
SETLOVDD  TCY       DL9       PREPARE FOR BATTERY CHECK
          SETR
          CALLL     MEASURE   ADC RESULT -> M(XA/D,12-15)
          ACACC     15        AC # 0 IF ERROR
          CALL      WARNING   USER DEFINED ERROR ROUTINE
          TCY       12        STORE RESULT TO M(10,12-15)
L$010     LDX       XA/D
          TMA                 M(XA/D,12-15) -> M(10,12-15)
          TAMIYC
          YNEC      0
          BRANCH    L$010
          TCY       DL9       ALLOW A1 TO A4 USAGE AGAIN
          RSTR
          RETN
*
* SUBROUTINE TESTS IF VDDMIN IS REACHED.
* THIS ROUTINE SHOULD BE CALLED REGULARELY (EG. ONCE PER DAY)
*
* CALL:   CALLL     TSTVDD
*         ACACC     15        TEST RESULT IN A
*         BRANCH    VDDLO     A # 0:  VDD =< VDDMIN
*                             A = 0:  VDD > VDDMIN
*
TSTVDD    TCY       DL9       PREPARE FOR BATTERY CHECK
```

```
          SETR
          CALLL     MEASURE   ADC RESULT -> M(XA/D,12-15)
          TCY       12        COMPARISON WITH M(10,12-15)
L$601     LDX       10        NREF: M(10,12-15)
          TMA
          LDX       XA/D      NMEAS: M(XA/D,12-15)
          ALEM
          BRANCH    L$600     NREFn =< NMEASn
          TCA       0         NREF > NMEAS:
          BR        L$603     VDD > VDDMIN
*
L$600     MNEA                NREFn =< NMEASn
          BRANCH    L$602     NREF < NMEAS: VDD < VDDMIN
          IYC                 NREFn = NMEASn
          YNEC      0         NEXT LOWER PART
          BRANCH    L$601     LSD YET HANDLED ?
L$602     TCA       15        NREF =< NMEAS: VDD < VDDMIN
L$603     TCY       9         ALLOW A1 TO A4 AGAIN
          RSTR                RESULT IN A
          RETN
```

6.12.2 Battery Check Routine II

The 1st subroutine is used during system calibration with $V_{DD} = V_{DDmin}$. The output of the measurement routine (only the value measured with noninverted inputs is used) is stored in M(10,13-15) for later references by the 2nd subroutine.

```
SETLOVDD  TCY       DL9       PREPARE FOR BATTERY CHECK
          SETR
          CALLL     MEASURE
          ACACC     15        AC # 0 IF ERROR
          CALL      WARNING   USER DEFINED ERROR ROUTINE
          LDX       10        STORE RESULT TO M(10,13-15)
          TCY       13
L$010     TDA                 M(15,13-15) -> M(10,13-15)
          TAMIYC
          YNEC      0
          BRANCH    L$010
          TCY       DL9       ALLOW A1 TO A4 USAGE AGAIN
          RSTR
          RETN
*
* SUBROUTINE TESTS IF VDDMIN IS REACHED. M(15,15,0) SHOWS
* RESULT. THIS ROUTINE SHOULD BE CALLED REGULARELY (EG. ONCE
* PER DAY).
*
* CALL:  CALLL      TSTVDD
*        TBIT       0          TEST RESULT M(15,15,0)
*        BRANCH     VDDLO      IF SET:  VDD < VDDMIN
*                              NOT SET: VDD > VDDMIN
*
TSTVDD    LDX       10         COPY M(10,13-15) -> M(15,13-15)
          TCY       13
L$011     TMA
          XDA
          IYC
          YNEC      0
          BRANCH    L$011
*
          TCY       DL9        PREPARE FOR BATTERY CHECK
          SETR
          SCSV                 SWITCH SVDD ON
          RCI                  SWITCH TO INVERTED COMP. INPUTS
          LDX       15         USE M(15,15,0) FOR COMPARISON
          TCY       15         RESULT:
          TCTM      0          M(15,15,0) = 0: VDD > VDDMIN
          TCY       9          M(15,15,0) = 1: VDD < VDDMIN
          RSTR
          RCSV
          TCY       15         PREPARE Y FOR MAINLOOP TEST
          RETN
```

6.13 Initialization Routine

The first program part checks if Warm start conditions are met: M(XORG, 0-3) contains check code >A50F. If this code is there, no actions are started. If this condition is not met, the normal initialization routine starts.

See 2.17 for description of Warm start and Cold start.

```
          ORG       1920       POWER UP POINT: PAGE 15 PC 00
INIT?     LDX       XORG       TEST CHECK CODE M(XORG,0-3)
          TCY       CHCKCOD
          TCA       10
          MNEA                 >A
          BRANCH    INIT
          ACYY       1
          BR         INIT2      SKIP WAKEUP POINT
*
          ORG        1927       WAKEUP POINT: PAGE 15 PC >7F
          ....
          ....
*
INIT2     TCA        5
          MNEA                  5
          BRANCH     INIT
          ACYY       1
          MNEZ                  0
          BRANCH     INIT
          ACYY       1
          TCA        15
          MNEA                  >F
          BRANCH     INIT
          DONE                   RAM IS OK: SPURIOUS INIT-SIGNAL
*
* RAM CONTENTS ARE DESTROYED: INITIALIZE TSS400
*
INIT      LDX        1          CLEAR M(1,0-15) AND M(9,0-15)
          CALL       CLRRAM
          LDX        2          X: 2 UND A
          CALL       CLRRAM
          LDX        3          X: 3 UND B
          CALL       CLRRAM
          LDX        4          X: 4 UND C
          CALL       CLRRAM
          LDX        7          CLEAR DAM
          CALL       CLRRAM
*
* NOW INITIALIZE THE NOMINAL CONSTANTS FOR COMPUTING
*
          LDX        XCONST
          TCY        YB1        B1: 1.000
          TCMIY      1
          ...

          LDX        XORG       SET CHECK CODE NIBBLES TO
          TCY        CHCKCOD    >A50F
          TCMIY      10
          TCMIY      5
          TCMIY      0
          TCMIY      15
          BL         MAINLOOP   START PROGRAM
*
```

```
* SUBROUTINE CLEARS ADDRESSED RAM BANK UND RAM BANK+8
*
CLRRAM    CLA
          TCY       15
CLRLP     TAMZA               0 -> M(X,Y)
          COMX8
          TAMDYN              0 -> M(X+8,Y)
          BRANCH    CLRLP
          RETN
```

6.14 Waiting Routines

It is often necessary to wait a certain or minimum time before the processing may continue. 3 subroutines for different delay time magnitudes are shown. The delay time of each subroutine can be computed by the formula:

$$T_{delay} = \frac{6}{F_{osc} \cdot \text{instruction count}}$$

If the delay time is only few microseconds, the following subroutine with several entry points may be used:

```
DELAY7    MNEA      7 INSTRUCTIONS DELAY INCL. CALL
DELAY6    MNEA      6 INSTRUCTIONS DELAY
DELAY5    MNEA      5 INSTRUCTIONS DELAY
DELAY4    MNEA      4 INSTRUCTIONS DELAY
DELAY3    MNEA      3 INSTRUCTIONS DELAY
          RETN      2 INSTRUCTIONS DELAY
```

If the delay time is in the range of milliseconds the above mentioned subroutine would waste to much ROM space. The following subroutine uses two nested loops. The delay time can be computed by the formula:

$$T_{delay} = \frac{6}{F_{osc} \cdot (3 + ((A + 1) \cdot 2 + 2) \cdot (Y + 1))}$$

```
* SUBROUTINE FOR WAITING 2 MS (FOSC = 500 KHZ)
*
WAIT2MS   TCY       6
WAITXMS   TCA       10        ENTRY POINT FOR SET Y REGISTER
L$804     ACACC     15
          BRANCH    L$804
          DYN
          BRANCH    WAITXMS
          RETN
```

To achieve other delays than 2 milliseconds, the Y register can be loaded with other values and the entry point WAITXMS can be used. Note the offset +1 which is added to the accumulator and the Y register in the formula.

```
* SUBROUTINE FOR WAITING 4 MS (FOSC = 500 KHZ)
*
WAIT4MS   TCY       13         (13 + 1) = 2 x (6 + 1)
          BR        WAITXMS
```

The maximum delay which is possible with the above subroutine is 547 instructions (6.56 ms at F_{OSC} = 500 kHz). It is achieved if accumulator and Y register are loaded with 15. If the needed delays exceed this magnitude, the timer of the TSS400(/4) can be used: The timer is read and a number added to the read value. When the timer reaches this computed value, the delay time elapsed. The uncertainity is 1 timer step (1/16 s). If Timer0 is used, the delay steps are seconds instead of 1/16 seconds.

```
WAIT      TCY       DL12        ADDRESS TIMER1 (OR TIMER0)
          SETR
L$200     TTA                   READ STABLE TIMER
          TNEA
          BRANCH    L$200
          ACACC     N           ADD N x 1/16 SECOND
          TAY                   AND STORE TO Y FOR COMPARISON
L$201     TTA                   READ STABLE TIMER AGAIN
          TNEA
          BRANCH    L$201
          YNEA                  COMPARE TIMER WITH
          BRANCH    L$201       COMPUTED VALUE
          RETN                  DELAY TIME ELAPSED
```

APPENDIX A

Differences of the TSS400/4 compared to the TSS400

The TSS400/4 is the 4 K-version of the TSS400. The architecture of both is nearly the same, only few differences exist due to the ROM and RAM-enlargement. These differences are listed below.

A.1 ROM Addressing

The ROM of the TSS400/4 has 4096 words of 10 bits each. The ROM is organized into 16 pages of 256 words each (16 · 256 = 4096 words total).

A.2 Program Counter Stepping Order

The 256 words of a page make a different stepping order necessary. The actual stepping order is shown below. The first half of it is identical to the TSS400 stepping order, the second half contains the same order with the MSB set to one.

```
-->
00 01 03 07 0F 1F 3F 7F 7E 7D 7B 77 6F 5F 3E 7C
79 73 67 4F 1E 3D 7A 75 6B 57 2E 5C 38 70 61 43
06 0D 1B 37 6E 5D 3A 74 69 53 26 4C 18 31 62 45
0A 15 2B 56 2C 58 30 60 41 02 05 0B 17 2F 5E 3C
78 71 63 47 0E 1D 3B 76 6D 5B 36 6C 59 32 64 49
12 25 4A 14 29 52 24 48 10 21 42 04 09 13 27 4E
1C 39 72 65 4B 16 2D 5A 34 68 51 22 44 08 11 23
46 0C 19 33 66 4D 1A 35 6A 55 2A 54 28 50 20 40
80 81 83 87 8F 9F BF FF FE FD FB F7 EF DF BE FC
F9 F3 E7 CF 9E BD FA F5 EB D7 AE DC B8 F0 E1 C3
86 8D 9B B7 EE DD BA F4 E9 D3 A6 CC 98 B1 E2 C5
8A 95 AB D6 AC D8 B0 E0 C1 82 85 8B 97 AF DE BC
F8 F1 E3 C7 8E 9D BB F6 ED DB B6 EC D9 B2 E4 C9
92 A5 CA 94 A9 D2 A4 C8 90 A1 C2 84 89 93 A7 CE
9C B9 F2 E5 CB 96 AD DA B4 E8 D1 A2 C4 88 91 A3
C6 8C 99 B3 E6 CD 9A B5 EA D5 AA D4 A8 D0 A0 C0
```

Figure A-1: TSS400/4 PC Stepping Order

A.3 RAM Addressing

The RAM of the TSS400/4 consists of 15 banks each containing 16 four-bit words. The legal operands of the LDX instruction are 0 to 15, where 7 and 15 address the same RAM bank: The DAM.

A.4 Opcodes

The BRANCH and CALL instructions have new opcodes due to the larger pages. All other instructions have unchanged opcodes.

Mnemonic	TSS400 Opcodes	TSS400/4 Opcodes
BRANCH	>100 - >17F	>200 - >2FF
CALL	>180 - >1FF	>300 - >3FF
All other Opcodes	>000 - >0FF	>000 - >0FF
Disabled	---	>100 - >1FF

A.5 MOS Oscillator Frequency

The MOS-oscillator frequency is 1 MHz ± 20 %. This tighter tolerance eases the "worst case design" of time critical software parts and reduces the current consumption of the measurement part.

TSS400 Family User's Guide

Sensor Signal Processors

TSS400 Family User's Guide

Sensor Signal Processors

Part IV

TSS400 Family User's Guide

Sensor Signal Processors

TSS400 STANDARD USER'S GUIDE

November 1990

Important Notice

Texas Instruments (TI) reserves the right to make changes in the devices or the device specifications identified in this publication without notice. TI advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

In the absence of written agreement to the contrary, TI assumes no liability for TI applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does TI warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Abstract

The TSS400 Standard User's Guide describes the application of a preprogrammed TSS400. The user's program is written in an interpreter language which is read and executed by an interpreter contained in the TSS400 ROM. This interpreter language allows nearly the direct conversion of a flowchart to written code. The user's program is contained in an EEPROM connected to the TSS400. The software routines included in the ROM are explained in detail. Examples are used wherever appropriate. The instruction set of the interpreter language is described in depth.

Table of Contents

List of Figures

List of Examples

1 Introduction

1.1 General

This section introduces the TSS400 Standard and outlines how an algorithm is developed and implemented to achieve cost effective designs. This introduction includes a definition of terms and conventions. This manual treats the TSS400 Standard as a system of logic and analog blocks controlled by the user.

After the introduction of the hardware a detailed presentation of the interpreter's instruction set will follow. Hints for efficient algorithms and a number of example programs are presented in the last sections, since they require a thorough understanding of the interpreter's instruction set.

The development tools are described in the last chapters.

1.2 Design Features

The TSS400 Standard is developed to suit a wide variety of applications where the costs of a ROM mask are not appropriate. Instead proven software routines are stored in the TSS400 ROM which can be called according to the user's need with an interpreter contained in the ROM, too. The user's program is stored in external EEPROMs which are loaded with interpreter opcodes by the development tool.

Features

- Minimum component count: System consists of 2 devices only, in a minimum configuration:
  - TSS400 with preprogrammed interpreter software
  - EEPROM with user's program
- 512 instructions (8 bits wide) in each EEPROM. Up to four EEPROMs are possible (2048 byte maximum). There is also an EEPROM available holding 2048 byte in one package.
- 12 bit A/D converter with 4 selectable inputs (A1 to A4)
- A/D converter range 0.1013 to 0.4946 · SV_{DD}

- Programmable current source from 0.15 to 2.4 mA · V_{DD}/V for all A/D converter inputs.
- LCD driver with 7 digits using 4 common multiplex.
- 64 preprogrammed characters, contained in a 7 segment output PLA. The decimal point, segment H, is switched independently.
- 576 bits of static RAM
- Three levels of subroutine
- Timekeeping from an on-chip cristal oscillator (32768 Hz)
- Done Mode for reduction of power consumption with active timers and RAM. Wake-up is possible by 2 timers (input frequencies 1 Hz and 16 Hz) or by input changes.
- Off Mode for very low power consumption with active RAM. Wake-up by input changes only.
- Internal MOS oscillator: 500 kHz to 1 MHz
- 6 freely usable push-pull outputs (R1 to R6)
- 4 bit I/O port (K-Port)
- Program control by EEPROM or host computer (Slave Mode)

1.3 Design Steps

Several steps are necessary to complete a system consisting of the TSS400 Standard and its peripheral with the necessary performance. A typical and recommended development cycle is shown below:

1. Definition of the tasks to be performed by the TSS400 and its peripheral.
2. Worst case timing consideration for the tasks.
3. Drawing of a complete hardware schematic.
4. Worst case design for all external components.
5. Organization of variables: Storage is possible in six Storage Registers and in EEPROM bytes.
6. Flowcharting of the complete program for determining the instruction coding necessary to fulfil the specification requirements.
7. Coding of the program with an editor.
8. Assembling of the program with the ASM400 assembler.
9. Removing of errors found by the ASM400 assembler.

10. Testing of the program with the ADT400 emulator or the Software Simulator.

1.4 Symbols and Conventions

1.4.1 List of Abbreviations

ADC	Analog Digital Converter
ALU	Arithmetic and Logic Unit
CPU	Central Processing Unit
DLn	Digit Latch n (n = 0 - 15)
EEPROM	Electrically Eraseable Programmable Memory
FLAC	Register used for Arithmetic and Results
Kn	K-Port n Pin (n = 1, 2, 4, 8)
LCD	Liquid Crystal Display
LSB	Least Significant Bit
LSD	Least Significant Digit
MSB	Most Significant Bit
MSD	Most Significant Digit
OPLA	Output Programmable Logic Array
PC	Program Counter (Interpreter)
RAM	Random Access Memory (Read/Write)
ROM	Read Only Memory
REGB	Register for arithmetic Operands
Rn	R Output n (n = 0 - 7)
W	Branch Address of Instruction Field (11 bits)
SRn	Subroutine PC Register Level n (n = 1 - 3)
STOn	Storage Register n (n = 0 - 5)
POS	Positive Flag in Status Bits
NEG	Negative Flag in Status Bits
ZERO	Zero Flag in Status Bits
10En	10 raised to n-th Power
EMI	Electro Magnetic Interference

1.4.2 Symbols and Logic Notations

a → b	Transfer Value a to b
a ↔ b	Exchange Contents of a and b
>NNN	NNN in hexadecimal Number Format
NNN	NNN in decimal Number Format
|NNN|	Absolute Value of NNN in decimal Number Format
=	Equal
#	Not equal
>	Greater than
≥	Greater than or equal to
<	Less than
≤	Less than or equal to
+	Addition
-	Subtraction
·	Multiplication
x	Don't Care Condition
:	Division
.OR.	Boolean OR Function
.AND.	Boolean AND Function
$\overline{X}$	One's Complement of X
XXX.YY	Number with 2 Digits after the decimal Point.
n	Number in Operands. Range defined by Instruction.

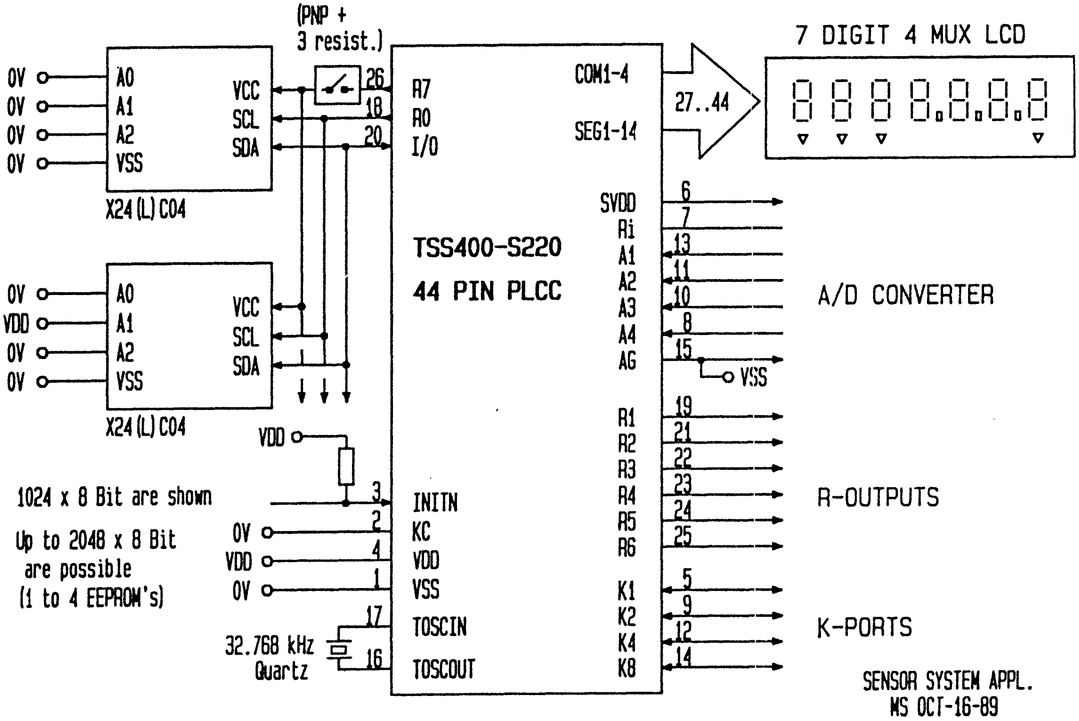


Figure 1: Blockdiagram of the TSS400 Standard

2 Hardware and Operation

2.1 General

As shown in Figure 1 up to four EEPROMs maybe connected to the TSS400 Standard. The TSS400 contains an interpreter in its ROM which uses the data storen in the EEPROM(s).

The interpreter reads the next instruction from the EEPROM, interprets the read opcode and executes the function defined by the opcode. If the read opcode needs much time for execution (e.g. multiplication) the supply voltage of the EEPROM is switched off for power saving. The instruction timing consists of the instruction fetch time and the execution time . During initalization, the interpreter's Program Counter PC is reset to location >000. Afterwards the Program Counter counts to the next address in a binary sequence.

The user's program is stored in one or more EEPROMs. It is written in a language ranging between assembler and compiler: The TSS400 Standard interpreter language. The interpreter language is assembled with the ASM400 assembler (see chapter 8 for details). The object created by the ASM400 assembler is transferred by the host computer via utility programs to the EEPROMs.

The EEPROMs are connected to the TSS400 via 4 lines:

1. R0 clocks the EEPROM during data transfers
2. R7 controls a PNP transistor which switches the EEPROM supply voltage
3. The I/O Pin transfers the data to and from the EEPROM
4. Common ground line

The 3 first mentioned lines cannot be used by the user's program due to by the interpreter usage.

2.2 EEPROM Addressing

The used EEPROM X24C04 handles 512 words each 8 bit which contain the user's program. If more than 512 bytes of program are necessary, the address space can be enlarged up to 2048 bytes by paralleling of EEPROMs. If more than two EEPROMs are necessary, the use of an X24C16 is considerable. It replaces four X24C04 with one package. The hardware address of the 4 EEPROMs is defined by the logic levels of their pins A1 and A2 as follows:

EEPROM Pins	A2	A1	Address Space
	0	0	0 - 511 (>0 → >1FF)
	0	1	512 - 1023 (>200 → >3FF)
	1	0	1024 - 1535 (>400 → >5FF)
	1	1	1536 - 2047 (>600 → >7FF)

Figure 2: Adressing of the EEPROMs

The register used to address the EEPROM is the Program Counter PC which is 11 bits wide. This allows the addressing of 2048 bytes without paging.

2.3 Jumps

Jumps are unconditional or conditional. An unconditional jump (JMP) is always executed successfully. A conditional jump (Jcond) depends on certain Status Bits. A successfully executed jump is defined to be the case when the jump transfers control to an instruction address out of the normal sequence. This is done by transferring the 11 bit address W contained in the opcode of the jump command into the Program Counter PC. An unsuccessful jump does not affect the normal sequence of the Program Counter.

Jump commands:

JMP	Jump unconditionally
JZ/JEQ	Jump if the ZERO Bit is set

JNZ/JNE	Jump if the ZERO Bit is reset
JP	Jump if the POS Bit is set
JN	Jump if the NEG Bit is set

2.4 Subroutines

Three levels of subroutine nesting are allowed. A 4th level CALL does not work properly. A subroutine call initiates the following activities:

- The subroutine stack is lowered one level and the return address is pushed onto the stack (SR1). (The return address is the address after the CALL instruction and is called normally PC + 2).
- The address field W of the CALL instruction is written into the Program Counter PC.

When a return instruction RETN occurs in a subroutine:

- The Subroutine PC Register SR1 (which holds the address after the last CALL instruction) is written to the PC.
- The stack is raised one level.

Warning: If the RETN instruction is used outside of a subroutine the last Subroutine Stack item is transferred to the PC. The program will not work properly.

Note: When considering subroutine stack levels it should be remembered that the instructions MOVPRMRB and MOVFLPRM also use one subroutine stack level (see later).

2.5 RAM Usage

There are 9 RAM banks, each containing 16 four-bit words (nibbles) in the RAM's 576 bit matrix. Eight of these RAM banks are normal RAM banks, one of them is a special Direct Access Memory (DAM). A complete RAM map is shown in chapter 8.

The TSS400 Standard RAM is organized in several registers, which are described in the following if they are visible to the user. Two RAM banks are used for supervising purposes, subroutine stack, PC and so on.

2.5.1 The FLAC Register

The FLAC register (the name is taken from the used Integer Math Package) is the main working register. It consists of eight nibbles containing the number itself and a ninth nibble which holds the sign and 2 flags used by the arithmetic routines. For positive numbers the sign bit is zero, for negative numbers the sign bit is one.

								SIGN
10E7	10E6	10E5	10E4	10E3	10E2	10E1	10E0	
MSD							LSD	SIGN

Figure 3: FLAC Register

The FLAC is used for several purposes:
- stores the result of all arithmetic and logic operations
- holds the 1st operand for arithmetic and logic operations
- contains the result of a hexadecimal to decimal conversion
- holds the information for transfers to the EEPROM
- holds the information to be displayed in the LCD

Instructions which affect or use the FLAC are:

LDFLPOS	load FLAC with a positive number
LDFLNEG	load FLAC with a negative number
MOVFLSTO	move FLAC to Storage Register
MOVSTOFL	move Storage Register to FLAC
MOVFLPR	move FLAC to EEPROM
MOVFLRB	move FLAC to REGB
MOVRBFL	move REGB to FLAC
CLRFL	clear FLAC

EXCHRBFL	exchange FLAC and REGB
HEXDEC	convert hexadecimal number to decimal
ADD	add REGB to FLAC
SUB	subtract REGB from FLAC
MPY	multiply REGB and FLAC
DIV	divide REGB into FLAC
ADDH	add REGB to FLAC hexadecimally
SUBH	subtract REGB from FLAC hexadecimally
AND	logically AND FLAC and REGB
OR	logically OR FLAC and REGB
CMPFLRB	compare FLAC and REGB
ROUNDn	round FLAC n times
SHIFTRn	shift FLAC right n times
SHIFTLn	shift FLAC left n times
CHKBATT	battery check with ADC value in FLAC
CHKCOMP	analog input compared with ADC value in FLAC
ADJBATT	battery measurement with the internal reference voltage
ADJCOMP	ADC value of a single measurement to FLAC
DISPLFL	display the FLAC as defined in the following bytes
KIN	K-Port to FLAC's LSD and to Group 1 Flags

Note: The FLAC will never contain -0 after an arithmetic operation. The status setting routine will always place +0 into the FLAC if all digits are equal to zero.

2.5.2 The REGB Register

The REGB register (the name is taken from the used Integer Math Package) is the second working register. It consists of eight nibbles containing the number itself and a ninth nibble which holds the sign. The sign bit is zero for positive numbers and one for negative numbers. See Figure 3 for the structure of the REGB, which is the same one as for the FLAC.

The REGB is used for:
- holding the 2nd operand for arithmetic and logic operations
- holding the constants read from the EEPROM
- holding the contents after transfers of the counters

Instructions which affect or use the REGB are:

LDRBPOS	load REGB with a positive number
LDRBNEG	load REGB with a negative number
MOVRBSTO	move REGB to Storage Register
MOVSTORB	move Storage Register to REGB
MOVFLRB	move FLAC to REGB
MOVRBFL	move REGB to FLAC
MOVCNTn	move Counter n to REGB
MOVPRMRB	move EEPROM to REGB
CLRRB	clear REGB
EXCHRBFL	exchange FLAC and REGB
HEXDEC	REGB is destroyed by the conversion
CMPFLRB	compare FLAC with REGB
TSTRB	test REGB
	all arithmetic and logic instructions

2.5.3 The Storage Registers

For storing information 6 Storage Registers may beused. They are configured exactly like the FLAC with 8 digits and a sign nibble. See Figure 4 for details. Storage Registers 1 to 5 hold their information until they overwritten by the MOVFLSTO or MOVRBSTO instructions.

The Storage registers are organized as follows

STOn	10E7	10E6	10E5	10E4	10E3	10E2	10E1	10E0	SIGN
	MSD							LSD	SIGN

Figure 4: Storage Registers STO0 to STO5

Storage Register 0 (STO0) is restricted in its use because it is used by the multiplication and division, too. This means it can hold information only until the next MPY, DIV or HEXDEC instruction occurs. After a multiplication it is set to 0, after a division it contains the remainder of the operation. This remainder may be used in conversions e.g. minutes to hours. See the Clock Routine example in chapter 7.

Instructions which use the Storage Registers:

MOVFLSTO	move FLAC to Storage Register
MOVSTOFL	move Storage Register to FLAC
MOVSTORB	move Storage Register to REGB
MOVRBSTO	move REGB to Storage Register

2.5.4 The Flag Registers

For common purposes two flag registers exist which contain 16 flags each. They are named Group 1 and Group 2. The selection is made by the instructions SELGRP1 and SELGRP2. The group in use is addressed until the other group is selected. These flags may be set, reset or tested. The usage of the 32 flags is shown below:

Group 1	Flag	0	Arbitrary use
	Flags	1 to 7	Segment H info for digits 1 to 7
	Flags	8 to 9	Arbitrary use
	Flag	10	LT: Long Timer reached zero if set
	Flag	11	ST: Short Timer reached zero if set
	Flags	12 to 15	K1 to K8 buffer
Group 2	Flags	0 to 15	Arbitrary use

Group 1

H3	H7	ST	K8
H2	H6	LT	K4
H1	H5	9	K2
0	H4	8	K1

Group 2

3	7	11	15
2	6	10	14
1	5	9	13
0	4	8	12

Figure 5: Flag Register Groups

Instructions which affect or use the flags are:

SBIT	set the addressed flag to one
RBIT	reset the addressed flag to zero
TBIT	test the addressed flag
KIN	read information from K-Port to Group 1 Flags 12 - 15
KINTIM	ead information from K-Port to Group 1 Flags 12 - 15
SELGRP1	select Group 1 flags
SELGRP2	select Group 2 flags
LDTIMS	reset Group 1 Flag 11 and load Short Timer
LDTIML	reset Group 1 Flag 10 and load Long Timer
STPTIMS	reset Group 1 Flag 11 and stop Short Timer
STPTIML	reset Group 1 Flag 10 and stop Long Timer

The Flags 1 to 7 in Group 1 control the H-segments of the display. If a H-flag is set, the according H-segment (decimal point) will be active. If H-flag is reset, the according H-segment will be off, except a display instruction switched it on by setting the H-bit contained in the according operand byte.

The Flags 10 and 11 in Group 1 are set by the timers when they reached zero. Resetting is to be done by the user's software.

The Flags 12 to 15 in Group 1 contain the K-Port information read in by the instructions KIN and KINTIM.

2.5.5 The Counters

Two decimal counters, Counter 1 and Counter 2, which can be loaded with constants ranging from 0 to 99 are included. They may be connected together forming a counter ranging up to 9999. Counter 1 is the least significant part of this combined counter. After the incrementing or decrementing of the counters the ZERO Bit is set or reset according to the result.

For computing or comparing purposes the counters may be transferred to the LSDs of the REGB. This may be done for the single or the combined counters. Decimal counting is used to ease computations with the counter contents after such transfers.

Instructions which affect or use the counters:

LDCNT1	load Counter 1 with the constant in the byte following
LDCNT2	load Counter 2 with the constant in the byte following
INCCNT1	increment Counter 1
INCCNT2	increment Counter 2
DECCNT1	decrement Counter 1
DECCNT2	decrement Counter 2
INCDBL	increment the combined counters
DECDBL	decrement the combined counters
MOVCNT1	move Counter 1 to REGB
MOVCNT2	move Counter 2 to REGB
MOVDBL	move the combined counters to REGB

2.6 The R Outputs and Digit Latches

The TSS400 contains of one-bit latches which can be set and reset independently: The Digit Latches DLn.

Two groups of Digit Latches can be distinguished:

- R outputs: The latches addressed from one to six are connected to external outputs named R1 to R6. All of them are equal in function. They can be used for several purposes:
  - Scanning of a keyboard, switches or programming diodes. The TSTKEY instruction uses R1 to R4 for this purpose.
  - Controlling of relays, lamps, LEDs etc. via buffers.
  - Switching of gates, multiplexers etc. directly.

Note: The R-outputs R0 and R7 cannot be addressed due to their usage by the control software. R0 steers the clock and R7 switches the supply voltage of the EEPROM. The assembler indicates an error message if one of these outputs is addressed.

- Digit Latches without external outputs: These latches are addressed by the operands eight to thirteen. Each Digit Latch DLn has a unique function as described below:
  - DL8: Controls the direction of the K-Port:
    DL8 = 0: The K-Port is an input port.
    DL8 = 1: The K-Port is an output port.
  - DL9 - DL11: Control of the analog multiplexer of the ADC:

DL11	DL10	DL9	Function
0	0	0	A1 to comparator
0	1	0	A2 to comparator
1	0	0	A3 to comparator
1	1	0	A4 to comparator
x	x	1	Battery check

Figure 6: Analog-Digital-Converter Addressing

Note: The Digit Latches DL9, DL10 and DL11 are controlled by the ADC instructions too. It is normally not necessary to change them by the user's software.

- DL12: Controls which timer register is connected to the ALU for the timer instructions and functions:
  DL12 = 0: 1 Hz input
  DL12 = 1: 16 Hz input
- DL13: Controls the constant current source of the ADC:
  DL13 = 0: Constant current source is off.
  DL13 = 1: Constant current source is on if SV_{DD} is on.

Instructions which affect the Digit Latches:

SETR	set the addressed Digit Latch to one
RSTR	reset the addressed Digit Latch to zero
TSTKEY	the addressed R output is used for keyboard scanning
MEASR	the ADC input addressing affects DL9, DL10, DL11

Initialization resets all Digit Latches to zero. The contents of the Digit Latches are retained during the Done Mode and the Off Mode as long as the supply voltage is maintained.

2.7 Status Logic

The status logic consists of 3 bits which are modified after the execution of several instructions. Within each instruction description the way the Status Bits are affected is explained. The Status Bits are checked by conditional jumps which are executed or not depending on the state of the tested Status Bit.

Note: Not every instruction influences the Status Bits. If an instruction does not affect the Status Bits, the status of the last instruction influencing the Status Bits is stored.

POS	NEG	ZERO

Figure 7: Status Logic Bits

2.7.1 Zero Bit

The bit ZERO is set to one if the result of the last instruction is zero. If the result is not zero the bit is cleared. The ZERO bit also indicates equality after comparisons in case it is set.

2.7.2 Positive Bit

The positive bit POS is set after arithmetic instructions if the result of an operation has a positive sign. If the result is negative the POS bit is reset. Other instructions will set the positive bit POS if no error occured. The cleared POS bit can be used as an error indication.

2.7.3 Negative Bit

The negative bit NEG is set after arithmetic instructions if the result of an operation has a negative sign. If the result is positive the NEG bit is reset. Other instructions will set the negative bit NEG if an error occured. The cleared NEG bit can be used as a "No Error" indication.

2.8 The K-Port

The K-Port is a four-bit I/O port. It's direction is controlled by DL8. The data to be output is stored in the four-bit K-Port Latch.

DL8 = 0: K-Port is switched to input direction
DL8 = 1: K-Port is switched to output direction. The contents of the K-Port Latch appears on the K-Port.

Because initialization resets DL8 the K-Port is switched to the input direction when initialization occurs.

The following instructions affect or use the K-Port:

TSTKEY	read the key connected to the addressed R-output
FLKOUT	output the LSD of the FLAC to the K-Port
KOUT	output a constant to the K-Port
KIN	read information from the K-Port
KINTIM	read information from the K-Port and update the timers

If DL8 is set the K-Port Latch is output to the K-Port. K8 is the MSB, K1 is the LSB. The contents of the K-Port Latch is not affected if DL8 is switched to input direction.

The K-Port is read via Schmitt-triggers into the ALU. Depending on the state of DL8, the input pins or the contents of the K-Port Latch is read in. The K-Port has no pull-down resistors.

The K-Port will wake-up the TSS400 from Done Mode and Off Mode under certain circumstances. See Figure 8 for the conditions for wake-up at the K-Port.

2.8.1 K8 wake-up Conditions

K1, K2 and K4 wake up the CPU only when changing from LO to HI. K8 differs from the other K-pins in its wake-up conditions. The CPU is woken up on every change of the K8-input if all other K-inputs are held LO. The interpreter checks the state of the K8 input each time Done Mode or the Off Mode is entered. It also prepares the wake-up logic to wake up in case the K8 input changes.

This feature is for applications which handle relatively high input frequencies. See Figure 8 for details.

2.9 The Timers

The TSS400 has an eight bit hardware timer register which is fed by a frequency derived from the cristal oscillator after dividing by 2048 (16 Hz from 32768 Hz). The timer register is used for updating the interpreter's two software timers: the Long Timer and the Short Timer.

Instruction which affect the Digit Latches:

SETR set the addressed Digit Latch to one

The cristal oscillator circuitry is intended for applications which need an accurate time base. The timer description is always related to an input frequency of 32768 Hz.

Two timers can be used:

- the Long Timer which counts up to 256 seconds
- the Short Timer which counts up to 16 (256/16) seconds

Note: It is the user's responsibility to set the Digit Latch DL12 to one if the Short Timer is used. If the Long Timer is used, only the state of DL12 is not of concern. The current consumption of the system is higher if DL12 is set.

The timers may be loaded with values up to 255 and are counted down with 1 Hz (Long Timer) resp. 16 Hz (Short Timer). When zero is reached or an underflow occurrs, the according flag in the Flag Group 1 is set to one and a wake-up occurs if Done Mode is active:

- Flag 10 (LT) is set when the Long Timer reaches zero
- Flag 11 (ST) is set when the Short Timer reaches zero

The two flags LT and ST are only set by the interpreter, never reset. Resetting of the flags must be done by the user's program with the instructions: LDTIML, LDTIMS, STPTIML, STPTIMS or RBIT. As long as one of the two flags is set, the program will leave the Done Mode immediately.

A timer which reached zero or run under zero is updated automatically by adding the last loaded value, which is stored in the Timer Buffer, to the actual contents of the timer. This assures stable timing as long as a underflow of the timer does not occur twice in a sequence. Two Timer Buffers exist:

- The Long Timer Buffer which stores the timer values loaded by the LDTIML instruction.
- The Short Timer Buffer which stores the values loaded by the LDTIMS instruction.

The Timer Buffers hold the loaded values until they overwritten by a new LDTIMx instruction. The loading of a Timer Buffer starts the according timer, which operates until a STPTIMx instruction is executed.

During Active Mode the timers have to be updated periodically with the instructions ACTTIM or KINTIM. This is done in time intervals ti. The conditions for ti are:

ti < loaded time by LDTIMS or LDTIML

and ti = < 15/16 s for the Short Timer

ti = < 15 s for the Long Timer

If a timer is stopped by a STPTIMx instruction the ACTTIM and KINTIM instructions do not affect it.

The timers may be used for several purposes:

- time base for software timers
- time base for software clocks
- measurement of time intervals
- keyboard debounce
- wait routines

The following instructions handle timer information:

LDTIML	load the Long Timer
LDTIMS	load the Short Timer
KINTIM	actualize the timers and the K-Port during Active Mode
ACTTIM	actualize the timers during the Active Mode
STPTIML	stop the Long Timer
STPTIMS	stop the Short Timer

2.10 The Current Saving Modes

It is not necessary for the TSS400 to stay on at all the time. The processor can be programmed to be active only if needed. This happens for updating the timekeeping or for performing software functions. The big advantage is the vast amount of current saved by using this feature. Two different modes of current saving are possible: the Done Mode and the Off Mode. The following part will describe the common elements of both modes:

Wake-up from initialization: see Figure 20 for conditions

Wake-up from K-Port: see Figure 8. Only if the conditions described under "before wake-up" are met, the conditions described under "wake-up condition" will wakeup the CPU from Done Mode or Off Mode.

For example if K8 = 0: Only if K1, K2 and K4 are also low, then the LO-HI change of any inputs will wake-up the TSS400.

K8		K4		K2		K1	Comment
0		0		0		0	before wake-up
1	.OR.	1	.OR.	1	.OR.	1	wake-up condition
1		0		0		0	before wake-up
0	.OR.	1	.OR.	1	.OR.	1	wake-up condition

Figure 8: Wake-up Conditions for external Events

The TSS400 Standard Registers wake up with the following contents:

Register	Done Mode	Off Mode
Program Counter PC	last PC + 1	last PC + 1
Status Bits POS NEG ZERO	unchanged	unchanged
RAM Contents	unchanged	unchanged
Digit Latches DLn	unchanged	unchanged
K-Port Latch Contents	unchanged	unchanged
Timers	actualized	undefined
ADC voltage SV_{DD}	switched off	switched off
LCD segment latches	unchanged	unchanged
Subroutine Stack	unchanged	unchanged

Figure 9: Wake-up Contents of the internal Registers

The wake-up address is always the next instruction following the DONE or OFF instruction (DONE or OFF work similar to an "IDLE" instruction). The unchanged Subroutine Stack allows a DONE or OFF instructions inside subroutines.

2.10.1 The Done Mode

DuringDone Mode the CPU is switched off completely. Only the timers, the RAM, the LCD driver, the Digit Latches and certain other registers are active. See Figure 9 for more information concerning the data retention during Done Mode. Other registers do not maintain information. Wake-up from this mode the timer, the K-Port or by linitiated by initialization.

Wake-up from timers: if the Long Timer or the Short Timer reaches zero the TSS400 is woken up. The wake-up from the timers occurs always, independent of the state of the K-Ports. The updating of the 2 timers is made without EEPROM activation. Only in the case zero is reached the EEPROM is switched on.

The Done Mode is entered by executing the DONE instruction. The ADC voltage SV_{DD} and the EEPROM supply voltage are switched when Done Mode is entered.

2.10.2 The Off Mode

During Off Mode the CPU is switched off completely. Only the RAM, the Digit Latches and certain other registers are active. See Figure 9 for more information concerning the data retention during Off Mode. Wake-up from Off Mode can be done by initialization or the K-Port only. See Figure 8 for wake-up conditions.

The current consumption is further reduced compared to the Done Mode.

The Off Mode is entered by executiing the instruction OFF. The ADC voltage SV_{DD} and the EEPROM supply voltage are switched off before OFF Mode is entered.

2.11 The Analog Digital Converter

The TSS400 contains an Analog Digital Converter (ADC) with a resolution of 12 bits. Four analog inputs are useable which are named A1 to A4. The input selection is made by the operand of the instruction MEASR:

MEASR	0	CONNECT A1 TO THE ADC
MEASR	1	CONNECT A2 TO THE ADC
MEASR	2	CONNECT A3 TO THE ADC
MEASR	3	CONNECT A4 TO THE ADC

Figure 10: Operand of the MEASR Instruction

The interpreter switches the internalDigit Latches DL9, DL10 and DL11 according to the operand of the MEASR instruction (see Figure 6).

The ADC does not measure absolute voltages but the ratio of the input voltage to the supply voltage. Hereby the measurement of resistive sensors is independent of the supply voltage. Absolute measurements are possible if SV_{DD} is held constant, which means a stable V_{DD} during a conversion and very small loading of SV_{DD}.

The following instructions control the A/D converter:

SVDDON	Switches SV_{DD} of the ADC on
SVDDOFF	Switches SV_{DD} of the ADC off
MEASR	Measures the addressed analog input

The instructions SVDDON and SVDDOFF are included only for special cases. Instructions using the ADC switch the SV_{DD} on and off automatically.

The following instructions use the A/D converter:

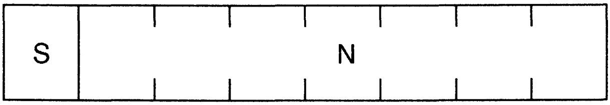
CHKBATT	battery check with the control value in the FLAC
CHKCOMP	analog input value compared with the value in the FLAC
ADJBATT	battery measurement with the internal reference voltage
ADJCOMP	transfers ADC value of a single measurement to the FLAC

2.11.1 Single and Compensated Measurements

Two different ADC measurements are possible:

- Single measurementsmade with the noninverted comparator inputs only. The offset error of the comparator is not compensated.
- Compensated measurements made by adding up the results of two measurements. One with noninverted comparator inputs and one with inverted inputs. The comparator's offset error is fully compensated this way.

Which kind of measurement is to be used, is defined in the byte following the MEASR instruction. This byte also defines how many single or compensated measurements have to be made:



S:	0:	Compensated measurements
	1:	Single measurements
N:	N+1	single or compensated measurements

Figure 11: Parameter Byte of the MEASR Instruction

2.11.2 Measurement Range and Conversion Formulas

The chosen range is the same for all inputs A1 to A4. The nominal properties of the ADC range are listed below (TSS400 User's Guide Revision 3.1):

N	A/D conversion result for a single measurement
Vmin	Vin/SV_{DD} for N = 1
Vmax	Vin/SV_{DD} for N = 4094 (>FFF - 1)
Resolution	delta in µV/SV_{DD} for 1 bit difference of result

ADC RANGE	V_{min}	V_{max}	Resolution
LARGE	0.101309	0.494607	96.1

The nominal conversion equations for single measurements are:

$$Vin/SV_{DD} = 0.101213203 + 0.000096090233 \cdot N$$
$$N = 10406.88496 \cdot Vin/SV_{DD} - 1053.314159$$

If more than one measurement result is added up, the above mentioned conversion equations have to be modified as follows:

$Vin/SV_{DD} = A + B \cdot N$	equation for a single measurement
$Vin/SV_{DD} = A + (B \cdot N)/M$	equation for M measurements added
$N = E \cdot Vin/SV_{DD} - F$	equation for a single measurement
$N = (E \cdot Vin/SV_{DD} - F) \cdot M$	equation for M measurements added

If the input voltage is below the lower limit Vmin of the ADC the value >000 is returned. If the input voltage is above the upper limit Vmax of the ADC the value >FFF is returned from the ADC. The NEG bit is set in both cases.

Note: The useable range of the Analog Digital Converter starts at values greater than 1 and ends at values less than >FFE due to the offset of the comparator. The useable range depends on the supply voltage. See the TSS400 User's Guide for actual values.

2.11.3 Battery Check

When DL9 is set, the analog inputs A1 to A4 are switched off from the ADC and the battery check connection is made: An internal reference voltage is connected to the ADC input and can be measured. Due to the ratiometric measurement principle of the ADC, the measured digital value is an indication of the supply voltage of the TSS400. The measured value is inversely proportional to the supply voltage V_{DD}. To get the reference for later battery tests a measurement is made with $V_{DD} = V_{DD}min$. The result is stored in the FLAC and can be transferred to a Storage Register or to the EEPROM. For later checks this value is transferred to the FLAC and compared to the internal reference voltage. The Status Bits POS or NEG are set according to the result.

The battery check feature is made by the following instructions:

ADJBATT	measure the internal reference voltage, result to FLAC
CHKBATT	compare the internal reference voltage to the FLAC.

2.11.4 The Current Source

A switchable current source is available for supplying special sensors. DL13 switches the current source which is simply programmable by an external resistor:

- DL13 = 0: current source is off
- DL13 = 1: current source is on if SV_{DD} is on

When switched on, the current source feeds the constant current out of the addressed (DL10, DL11) analog input An. The voltage generated at the external sensor is measured with the same analog input An.

The current I programmed by the external resistor Rext, which is connected between SV_{DD} and the input Ri, is given by the following equation:

$$I = V_{rext}/Rext$$

V_{rext} is approximatively $0.24 \cdot SV_{DD}$.

The current range for each ADC input is as follows:

$$0.15 \text{ to } 2.4 \text{ mA} \cdot SV_{DD}/V$$

This current range is sufficient to drive PT100 sensors too (Platinum temperature sensors with a nominal resistance of 100 Ohms).

2.12 The LCD Driver

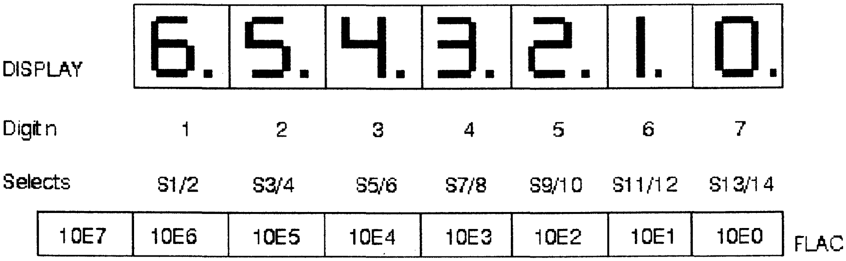
The TSS400 contains complex LCD driver circuitry which is defined for the TSS400 Standard in a way to get best results for a wide range of applications. From a software point of view the LCD driver looks very simple:

- No timing problems exist with multiplexing for getting a quiet, stable display. The LCD driver hardware outputs the loaded display information automatically, without any software burden, during Active Mode and Done Mode.
- The software has to decide which segment information has to be displayed at which digit only.

Instructions controlling the display:

DISPLCLR clear the whole display (blanks into all digits)
DISPLDGn load digit n with the following byte's info
DISPLFL load the display with the following info

The display used by the TSS400 Standard is configured like shown below. The according FLAC nibbles are shown too:



DISPLAY		6.	5.	4.	3.	2.	1.	0.
Digit n		1	2	3	4	5	6	7
Selects		S1/2	S3/4	S5/6	S7/8	S9/10	S11/12	S13/14
	10E7	10E6	10E5	10E4	10E3	10E2	10E1	10E0	FLAC

Figure 12: Connection of Display and FLAC

2.12.1 Digit Addressing

Normaly the contents of the FLAC is output to the display. The relations are as follows:

FLAC 10E0 → S13 + S14 Digit 7 LSD
FLAC 10E1 → S11 + S12 Digit 6
FLAC 10E2 → S9 + S10 Digit 5
FLAC 10E3 → S7 + S8 Digit 4

FLAC 10E4 → S5	+ S6	Digit 3	
FLAC 10E5 → S3	+ S4	Digit 2	
FLAC 10E6 → S1	+ S2	Digit 1	MSD

The FLAC's MSD (10E7) cannot be displayed normally due to the 7 digit configuration of the display driver. If it is necessary to display the MSD too, a right shift with decimal point correction needs to be done.

2.12.2 Segement Addressing

The OPLA terms definition is based on the following hardware configuration:

SELECT LINES							COMMON			
							1	2	3	4
S1	S3	S5	S7	S9	S11	S13	C	F	H	E
S2	S4	S6	S8	S10	S12	S14	A	B	D	G

Caution: The shown Common/Select definition cannot be modified. The used display **must** confirm to it.

Figure 13: Common/Segment Definition

This means for instance odd Select Lines and Common 4 address the segment E. The assumed segment notation follows:

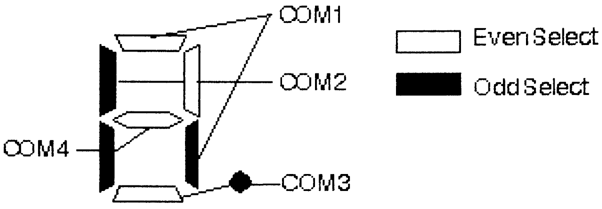


Figure 14: Standard Segment Notation

The chosen Common/Select configuration is fail safe. This means no valid numbers or characters are displayed in case of a segment or common failure. Instead senseless segment combinations are displayed.

2.12.3 The Output PLA

The TSS400 LCD driver contains a gate level Output PLA (OPLA) with 64 · 7 bits. The 7 bits represent the segment information A to G for 64 predefined combinations (the segment H info is independent from this OPLA and is given with the display instructions).

The addressing of these 64 segment combinations is made by the display instructions. See there for details. The character number in the following OPLA list is the number to be used with the instructions DISPLFL and DISPLDGn.

The OPLA block number is used if combinations of single segments are to be output by FLAC digits. The OPLA block number consists of two MSBs of the character number.

OPLA Block	Character Number	Segments	Character
0	0	A B C D E F	0
0	1	B C	1
0	2	A B D E G	2
0	3	A B C D G	3
0	4	B C F G	4
0	5	A C D F G	5
0	6	A C D E F G	6
0	7	A B C	7
0	8	A B C D E F G	8
0	9	A B C D F G	9
0	10	A B C E F G	A
0	11	C D E F G	b
0	12	A D E F	C or [
0	13	B C D E G	d
0	14	A D E F G	E

OPLA Block	Character Number	Segments	Character
0	15	A E F G	F
1	16	None	Blank
1	17	B C D	J
1	18	D E F	L
1	19	A B E F G	P
1	20	B C D E F	U
1	21	D E G	c
1	22	C E F G	h
1	23	D E	l
1	24	C E G	n
1	25	C D E G	o
1	26	E G	r
1	27	D E F G	t
1	28	C D E	v
1	29	B C D F G	Y
1	30	B C E F G	H
1	31	B C G	-1
2	32	None	Blank
2	33	A	
2	34	F	
2	35	A F	
2	36	B	
2	37	A B	
2	38	B F	
2	39	A B F	
2	40	G	Minus Sign
2	41	A G	
2	42	F G	
2	43	A F G	
2	44	B G	
2	45	A B G	
2	46	B F G	

OPLA Block	Character Number	Segments	Character
2	47	A B F G	Degree
3	48	E	
3	49	A E	
3	50	E F	
3	51	A E F	
3	52	B E	
3	53	A B E	
3	54	B E F	
3	55	A B E F	
3	56	E G	
3	57	A E G	
3	58	E F G	
3	59	A E F G	
3	60	C	
3	61	D	
3	62	A B C D	]
3	63	A B C D E	J

Figure 15: Characters contained in the OPLA

2.12.4 Special Display Considerations

If a display is needed which differs from the normal scheme (seven segments + one independent segment H) the characters contained in the OPLA have to be considered. Three additional possibilities exist beside the display of numbers with or without decimal point:

- single segment display
- bar graph display
- combinations of up to six segments

Single Segment Display:

For each single segment A to H a character number exists:

Character Number	Segments
33	A
36	B
60	C
61	D
48	E
34	F
40	G
16+H	H

Figure 16: Character Numbers for single Segments

Bar Graph Display:

A bar graph may be built with the complete segments A to H. The character numbers for each graph are shown below:

Character Number	Segments
33	A
37	AB
7	ABC
62	ABCD
63	ABCDE
0	ABCDEF
8	ABCDEFG
8+H	ABCDEFGH

Figure 17: Character Numbers for a Bar Graph Display

Combination of up to six Segments:

If a digit is used for the display of independent segments (signs, modes e.g.) the segments A F B G E and H may be used. The segments C and D may not be connected to this digit if all possible 64 terms are used. The combinations are structured starting at character number 32 up to 59. This means, the combinations may be output with a number in a FLAC digit with the correct OPLA block defined in the instructions parameter. The last 4 combinations have randomly defined character numbers and have to be output explicitly.

The six segments correspond to the following digit bits and OPLA blocks:

OPLA Block	Bit	Segments	Valence
2	0	A	LSB
2	1	F	LSB + 1
2	2	B	LSB + 2
2	3	G	MSB - 1
3	—	E	MSB
—	—	H	MSB + 1

Figure 18: Digit Contents of Segment Combinations

If a combination of segments A F B and G is needed, the appropriate bits have to be set in the addressed FLAC digit. The used OPLA block is 2 in this case. If segment E is needed too with segments A F B and G, the same bits in the addressed FLAC digit are set, but the OPLA block 3 is used. The segment H is possible for each combination by setting the H bit in the instruction's parameter. See the description of the instructions DISPLDGn and DISPLFL for details.

OPLA Block	Character Number	Segments						Digit Contents
2	32	—	—	—	—	—		0
2	33	—	—	—	—	A		1
2	34	—	—	—	F	—		2
2	35	—	—	—	F	A		3
2	36	—	—	B	—	—		4
2	37	—	—	B	—	A		5
2	38	—	—	B	F	—		6
2	39	—	—	B	F	A		7
2	40	—	G	—	—	—		8
2	41	—	G	—	—	A		9
2	42	—	G	—	F	—		10
2	43	—	G	—	F	A		11
2	44	—	G	B	—	—		12
2	45	—	G	B	—	A		13
2	46	—	G	B	F	—		14
2	47	—	G	B	F	A		15
3	48	E	—	—	—	—		0
3	49	E	—	—	—	A		1
3	50	E	—	—	F	—		2
3	51	E	—	—	F	A		3
3	52	E	—	B	—	—		4
3	53	E	—	B	—	A		5
3	54	E	—	B	F	—		6
3	55	E	—	B	F	A		7
3	56	E	G	—	—	—		8
3	57	E	G	—	—	A		9
3	58	E	G	—	F	—		10
3	59	E	G	—	F	A		11
0	13	E	G	B	—	—		13
0	2	E	G	B	—	A		2
1	30	E	G	B	F	—		14
1	19	E	G	B	F	A		3
1	19 + H	E	G	B	F	A	H	3

Figure 19: Character Numbers for Segment Combinations

If for example the combination F G (character number 42) is to be output, the addressed FLAC digit has to contain 10 (bit 3 and bit 1) and the OPLA block 2 has to be selected.

If additional the segment E is to be output with segments F and G, (character number 58), the addressed FLAC digit has to contain 10 again and the OPLA block 3 needs to be used. See also an example where of the display instructions are described.

2.13 Initialization and Powerup

Initialization is initiated by two hardware reasons:

- Power up: The voltage V_{DD} is switched on (Cold start). The CPU starts to work at PC 000 after the internal cristal oscillator started operation. This may last 1 to 6 seconds.
- INITN pin: If the INITN pin is held low for more than 10 µs (in case this occurs during program execution it is named Warm Start). The CPU starts operation at PC 000 when the INITN pin is released to V_{DD} potential.

The TSS400 Standard registers show the following contents after a power up or a INITN pin.

Register	Power up (Cold Start)	INITN pin (Warm Start)
Program Counter PC	000	000
Status Bits POS NEG ZERO	undefined	unchanged
RAM Contents (see Note)	reset to 0	unchanged
Digit Latches DLn	reset to 0	reset to 0
K-Port Latch Contents	undefined	unchanged
Timers	0	unchanged
ADC Voltage SV_{DD}	switched off	switched off
LCD Segment Latches	undefined	unchanged
Subroutine Stack	level 0	level 0

Figure 20: Initialization contents of the internal registers

Note: Despite the RAM remains unchanged if Warm Start (see below) occurs. The memory addressed when the INITN pin was activated may be destroyed by a write cycle.

If the TSS400 system is battery powered and contains calibration factors or other important data in RAM it is advisable to distinguish between Cold Start and Warm Start. The reason is the possibility of initializations caused by EMI. If such an erroneous initialization is not tested for egality, EMI influence could destroy the RAM contents by clearing the RAM by the initialization software routine. The TSS400 Standard compares two reserved RAM nibbles if they contain >A5 after each initialization:

- If the RAM nibbles contain expected right information >A5, continues at PC 000. The RAM contents is not influenced. This means a spurious signal caused the initialization (Warm Start).
- If the RAM nibbles differ from >A5, the RAM is cleared and the program continues at PC 000. This means the TSS400's supply voltage was switched on (Cold Start).

Note: The Short Timer and the Long Timer are not stopped by a Warm Start. This means, they stay active and have to be stopped by a STPTIMx instruction if necessary.

See Chapter 7 for an example how to distinguish the two start-ups by the user's software.

2.14 Hardware Options

The used hardware options are a compromise for getting best result for several applications. See the "TSS400 Software User's Guide" for a detailed description of the options.

2.14.1 Pinning

Pin	Connection	Pin	Connection
1	V_{SS}	23	R4
2	KC	24	R5
3	INITN	25	R6
4	V_{DD}	26	R7
5	K1	27	COM2
6	SV_{DD}	28	COM1
7	Ri	29	S14
8	A4	30	S13
9	K2	31	S12
10	A3	32	S11
11	A2	33	S10
12	K4	34	S9
13	A1	35	S8
14	K8	36	S7
15	AGND	37	S6
16	TOSCOUT	38	S5
17	TOSCIN	39	S4
18	R0	40	S3
19	R1	41	S2
20	I/O	42	S1
21	R2	43	COM3
22	R3	44	COM4

Figure 21: Pinning of the TSS400 Standard

2.14.2 Input Options

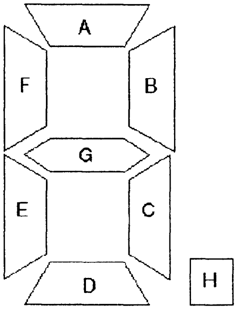
- K8 is connected directly to the ALU, the flip flop in the K8 logic is not active
- the K-Port has push pull outputs and no pulldown resistors
- the used ADC range is 0.1013 · SV_{DD} to 0.4946 · SV_{DD}
- the MOS oscillator speed is 500 kHz to 1 MHz

2.14.3 Timer Options

- 6 Hz are used for the wake-up frequency with set DL12
- no clock frequency is output to a pin
- the oscillator hardware is chosen for a 32768 Hz cristal

2.14.4 Display Options

- 4 common multiplex is used
- The following Common/Select configuration is used:
- The 64 possible characters are shown in Figure 15.



	COMMON			
	1	2	3	4
Odd Selects	C	F	H	E
Even Selects	A	B	D	G

Figure 22: Standard segment notation scheme

The complete hardware definition file of the TSS400 Standard is shown in Appendix A.

2.15 Instruction Execution Times

The complete instruction execution time consists of:
- Instruction fetch time from the EEPROM
- Instruction decoding time
- Instruction execution time by the TSS400

The exact timing of each instruction is very difficult to compute, caused by its dependance on circumstances. Measurements showed, that a good estimation value is 150 TSS400 instructions for each byte (not instruction!) read from the EEPROM. This means, if the oscillator frequency is 500 kHz, each byte of the user's code needs approximatively 1.8 ms for execution. ($150 \cdot 6/500000$ Hz = 1.8 ms).

This estimation bases on normal instruction sequences. If computation intensive instructions occur very often, the mean time of an instruction increases. The instruction times for these time consuming instructions are:

- Multiplication MPY:	average register contents	15 ms
- Division DIV:	average register contents	18 ms
- Conversion HEXDEC:	average register contents	46 ms
- Measurement MEASR:	Four compensated measurements	13 ms

With the above given average execution times, the time estimations should be corrected.

The program flow control instructions have the following execution times:

- CALL including RETN:	9.5 ms
- unconditional Jump:	4.5 ms
- conditional Jump executed:	4.5 ms
- conditional Jump not executed:	2.2 ms

The EEPROM instructions have the following execution times:

- EEPROM Write instruction:	MOVFLPRM	40 ms
- EEPROM Read instruction:	MOVPRMRB	15 ms

2.16 Usage of RAMS

The PHILIPS PCF8570, a $256 \cdot 8$ bit RAM can be used for TSS400 Standard applications due to the identical bus protocol. Often modified pointers for EEPROM addressing may be allocated inside of the RAM, which is not degraded by often occuring write operations. See also 7.9 ADDRESS MODIFICATION for more information.

The interpreter assumes blocks of 512 bytes as used with the X24C04 EEPROMs and issues therefore no addressing information if the 256 byte boundary of a RAM is exceeded. This means, the user has to include a jump instruction to get into the 2nd RAM in case it is used. Instructions with more than one byte may not be placed on the boundary due to that reason.

3 Interpreter Instruction Set Definitions

3.1 Interpreter Language Instruction Map

The instruction map is shown for convenience purposes. The numbers on the left margin show the MSD part of the opcode, the numbers above the map show the LSD part. The opcode for ROUND2 is E1 for example.

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th></tr>
<tr><td>F</td><td>FLKOUT</td><td>KOUT</td><td>KIN</td><td>SEL
GRP1</td><td>SEL
GRP2</td><td>MOV
CNT1</td><td>MOV
CNT2</td><td>MOV
DBL</td></tr>
<tr><td>E</td><td>ROUND
1</td><td>ROUND
2</td><td>ROUND
3</td><td>ROUND
4</td><td>SHIFT
R1</td><td>SHIFT
R"</td><td>SHIFT
L1</td><td>SHIFT
L"</td></tr>
<tr><td>D</td><td>LDTIML</td><td>LDTIMS</td><td>DONE</td><td>OFF</td><td>RETN</td><td>CLRFL</td><td>CLRRB</td><td>EXCH
RBFL</td></tr>
<tr><td>C</td><td colspan="8">MOVFLPRM</td></tr>
<tr><td>B</td><td>DISPL
FL</td><td>DISPL
DG1</td><td>DISPL
DG2</td><td>DISPL
DG3</td><td>DISPL
DG4</td><td>DISPL
DG5</td><td>DISPL
DG6</td><td>DISPL
DG7</td></tr>
<tr><td>A</td><td colspan="6">MOVFLSTO</td><td colspan="2">MOVSTOFL</td></tr>
<tr><td>9</td><td colspan="6">MOVRBSTO</td><td colspan="2">MOVSTORB</td></tr>
<tr><td>8</td><td colspan="8">JP</td></tr>
<tr><td>7</td><td colspan="8">JZ/JEQ</td></tr>
<tr><td>6</td><td colspan="8">JMP</td></tr>
<tr><td>5</td><td colspan="4">LDFLPOS</td><td colspan="4">LDFLNEG</td></tr>
<tr><td>4</td><td colspan="8">TBIT</td></tr>
<tr><td>3</td><td colspan="8">RBIT</td></tr>
<tr><td>2</td><td colspan="8">SBIT</td></tr>
<tr><td>1</td><td>ACTTIM</td><td colspan="6">RSTR</td><td>SVDD
OFF</td></tr>
<tr><td>0</td><td>NOP</td><td colspan="6">SETR</td><td>SVDD
ON</td></tr>
</table>

Figure 23: Instruction Map lower Part

	8	9	A	B	C	D	E	F
F	STP TIML	STP TIMS	SLV					(NOP)
E	ADJ BATT	ADJ COMP	CHK BATT	CHK COMP	CMP FLRB	TSTKE Y	TSTRB	KINTIM
D	DEC CNT2	DEC CNT1	DECDBL	INC CNT2	INC CNT1	INCDBL	LDCNT 2	LDCNT 1
C	MOVPRMRB							
B	DISPL CLR	MOV FLRB	MOV RBFL	ADD	SUB	MPY	DIV	HEX DEC
A	MOVSTOFL				MEASR			
9	MOVSTORB							
8	JN							
7	JNE/JNZ							
6	CALL							
5	LDRBPOS				LDRBNEG			
4	TBIT							
3	RBIT							
2	SBIT							
1	RSTR						SUBH	AND
0	SETR						ADDH	OR

Figure 24: Instruction Map upper Part

3.2 Effect on Status Bits

Each instruction description in this section contains a status description. The way in which an instruction depends on, or modifies the Status Bits is defined as follows:

Not affected:	the Status Bits remain unchanged
POS = 1	the Status Bit (here POS) is set
NEG = 0	the Status Bit (here NEG) is reset
ZERO = X	the Status Bit (here ZERO) remains unchanged

The information shown after STATUS is given for the completed instruction.

3.3 Interpreter Instruction Formats

The interpreter uses different instruction formats. Each instruction description shows the correct source format after the label MNEMONIC. The different source formats are explained in the following:

3.3.1 Constant Operand

```
INSTR n
```

The operand n, whose range is defined in the description of the instruction, is placed into the opcode's operand field or into the following byte by the assembler.

3.3.2 EEPROM Address Operand

```
INSTR LABEL
```

The operand, which is an EEPROM address, is placed into the opcode's LSBs and into a byte following the instruction by the assembler.

3.3.3 Implicit Operand

```
INSTR
```

The instruction does not have an operand.

```
INSTRn
```

The instruction contains the operand in the mnemonic. The range of n is defined in the instruction's description.

3.3.4 Hexadecimal Operand

```
INSTR >NN
```

The instruction uses hexadecimal or BCD numbers for operands. The assembler locates the operand in the next byte.

3.3.5 Decimal Operand

```
INSTR NNN
```

The instruction uses decimal numbers for operands. The assembler locates the operand in the next byte.

Note: Numbers maybe formatted >NN or NNN normally. Only if BCD numbers are used, the >NN format is necessary for separating the two digits.

3.3.6 Attached, Floating Number of Operands

```
INSTR n
BYTE  >NN
...
BYTE  >NN
```

The instruction uses n bytes following the instruction for additional information. The n bytes needs to be included into the source program by the user.

```
INSTR >NN
BYTE  >NN
...
BYTE  >NN
```

The number of additional bytes is defined by:

Operand LSD - MSD + 1.

3.3.7 Attached, Fixed Number of Operands

```
INSTR n
BYTE  >NN
```

The instruction uses one byte following the instruction. The byte is to be included by the user. An operand n is used, too.

3.3.8 Coding Format

The following rules should be followed in writing a program:

- Label fields are a maximum of eight alphanumeric characters starting with an alphabetic character. The label field begins in column one.
- The operation code (opcode) is to the right of a label, the two separated by at least one blank space. If no label is used, the operation code begins after the first column (second column or further right).
- The operand is to the right of the operation code, they are separated by at least one blank.
- A comment is to the right of the operand, they are separated by at least one blank. If a comment occupies a separate line, it must begin with an asterisk (*) in column one.

For legibility, it is recommended that fields begin in the following columns:

- label fields must begin in column 1
- operation codes should begin in columnn 11
- operands should begin in column 21
- comments to an instruction should begin in column 31
- comment lines must begin in column one with an asterisk

Example: The following lines show source lines conforming to the above rules.

```
* THIS IS A PURE COMMENT LINE
LABEL          OPCODE         OPERAND        COMMENT
|              |              |              |
|              |              |              column 31
|              |              column 21
|              column 11
column 1
```

3.4 Register to Register Transfer Instructions

3.4.1 Move FLAC to Storage Register

MNEMONIC: MOVFLSTOn OPCODE: A0 - A5

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC → STOn $(0 \leq n < 6)$

PARAMETER: none

PURPOSE: Copying or saving the FLAC into a Storage Register.

DESCRIPTION: The FLAC content is unconditionally transferred to the Storage Register n. The FLAC contents is unaltered.

3.4.2 Move Storage Register to FLAC

MNEMONIC: MOVSTOFLn OPCODE: A6 - AB

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: STOn → FLAC $(0 \leq n < 6)$

PARAMETER: none
PURPOSE: Reading a Storage Register.

DESCRIPTION: The content of the Storage Register n is unconditionally transferred to the FLAC. The content of the Storage Register n is unaltered.

3.4.3 Move REGB to Storage Register

MNEMONIC: MOVRBSTOn OPCODE: 90 - 95

STATUS:
If REGB = 0:
POS = 1 NEG = 0 ZERO = 1
If REGB > 0
POS = 1 NEG = 0 ZERO = 0
If REGB < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: REGB → STOn $(0 \le n < 6)$

PARAMETER: none

PURPOSE: Copying or saving the REGB into a Storage Register.

DESCRIPTION: The REGB content is unconditionally transferred to the Storage Register n. The REGB contents is unaltered.

3.4.4 Move Storage Register to REGB.

MNEMONIC: MOVSTORBn OPCODE: 96 - 9B

STATUS:
If REGB = 0:
POS = 1 NEG = 0 ZERO = 1
If REGB > 0
POS = 1 NEG = 0 ZERO = 0
If REGB < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: STOn → REGB $(0 \le n < 6)$

PARAMETER: none

PURPOSE: Reading a Storage Register.

DESCRIPTION: The content of the Storage Register n is unconditionally transferred to the REGB. The content of the Storage Register n is unaltered.

3.4.5 Move FLAC to REGB

MNEMONIC: MOVFLRB OPCODE: B9

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC → REGB

PURPOSE: Copying of the FLAC to REGB
DESCRIPTION: The content of the FLAC is transferred to the REGB. The Status Bits are set according to the FLAC content.

PARAMETER: none

EXAMPLE: The FLAC is to be multiplied by three:

```
...
MOVFLRB         MULTIPLY FLAC BY 3
ADD             x 2
ADD             x 3
...
```

Example 1: Multiplication by Three

3.4.6 Move REGB to FLAC

MNEMONIC: MOVRBFL OPCODE: BA

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: REGB → FLAC

PURPOSE: Copying of the REGB to the FLAC

DESCRIPTION: The content of the REGB is transferred to the FLAC. The Status Bits are set according to the FLAC contents.

PARAMETER: none

3.4.7 Exchange REGB and FLAC Registers

MNEMONIC: EXCHRBFL OPCODE: D7

STATUS: After execution:
If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: REGB ↔ FLAC

PURPOSE: Exchanging REGB and FLAC.

DESCRIPTION: The contents of REGB and FLAC are exchanged. The Status Bits are set according to the FLAC contents after execution of the instruction.

PARAMETER: none

3.4.8 Move FLAC to EEPROM

MNEMONIC: MOVFLPRM LABEL OPCODE: C0 - C7

STATUS: not affected

ACTION: PC+2 → Stack
FLAC → EEPROM
Stack → PC

PARAMETER: None. Two byte instruction.

PURPOSE: Storing of constants in the EEPROM.

DESCRIPTION: The content of the FLAC is written to four EEPROM bytes starting at the address defined by the label. This allows the storage of computed constants in the EEPROM which for example gives the possibility of self-calibration of the system. The sign of the FLAC is stored in the MSB of the number; this reduces the number's range to +/- 79999999. The MSD is stored at the first byte (High nibble of address LABEL). The inverse operation MOVPRMRB restores the sign correctly.

Note: One level of the stack is used for the storage of the PC. This means that this instruction cannot be used in the 3rd subroutine level.

EXAMPLE: The result of a computation need to be stored in the EEPROM address CONST2 for later use.

```
         ADD                   CONST2 COMPUTED IN FLAC
         MOVFLPRM    CONST2    WRITE CONST2 TO EEPROM
         ...
         ORG         500       CONSTANTS IN EEPROM ADDRESS 500
CONST2   BYTE        0         10E7/10E6 SIGN IN MSB
         BYTE        0         10E5/10E4
         BYTE        0         10E3/10E2
         BYTE        0         10E1/10E0    LSD
```

Example 2: Storage of a computed value in the EEPROM

The allocation inside the EEPROM is shown in Figure 25:

MSB		LSB	
S	10E7	10E6	500 ADDRESS
	10E5	10E4	501
	10E3	10E2	502
	10E1	10E0	503

Figure 25: EEPROM map of the MOVFLPRM values

3.4.9 Move EEPROM to REGB

MNEMONIC: MOVPRMRB LABEL OPCODE: C8 - CF

STATUS: not affected

ACTION:
PC+2 → Stack
EEPROM → REGB
Stack → PC

PARAMETER: None. Two byte instruction.

PURPOSE: Loading of constants stored in the EEPROM.

DESCRIPTION: The contents of the EEPROM address defined by the label and the contents of the following 3 bytes are written to REGB. The sign is restored out of the MSB of the number into the sign nibble of REGB. The MSD is stored in the 1st byte (high nibble of address LABEL).

Note: One level of the stack is used for the storage of the PC. This means that this instruction may not be used in the 3rd subroutine level.

EXAMPLE: The computed constant of a multiplication need to be stored in the EEPROM at the label CONST1. For later computations the constant is read back from the EEPROM:

```
A2        EQU        1
          ...
          MPY                   CONSTANT IN FLAC
          MOVFLPRM   CONST1     STORE CONSTANT IN EEPROM
          ...
          ...
          MEASR      A2         MEASURE A2 INPUT
          BYTE       1          WITH 2 COMPENSATED MEASUREMENTS
          HEXDEC                CONVERT ADC RESULT TO DECIMAL
          MOVPRMRB   CONST1     CONST1 -> REGB
          ADD                   ADD CONST1 TO ADC-VALUE
          ...
          ORG        504        CONSTANTS AREA IN EEPROM
CONST1    BYTE       0          MSDs AND SIGN
          BYTE       0
          BYTE       0
          BYTE       0          LSDs
          ...
```

Example 3: Storing and retrieving of an EEPROM value

3.5 Arithmetic Instructions

See Chapter 7 for useful hints and applications of these Instructions.

3.5.1 Add REGB to FLAC Decimally, Result to FLAC

MNEMONIC: ADD OPCODE: BB

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC + REGB → FLAC

PARAMETER: none

PURPOSE: Adding of two signed, decimal values.

DESCRIPTION: The content of REGB is added decimally to the content of the FLAC, the result is stored in the FLAC. REGB and STO0 are not altered.

3.5.2 Subtract REGB From FLAC Decimally, Result to FLAC

MNEMONIC: SUB OPCODE: BC

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC - REGB → FLAC

PARAMETER: none

PURPOSE: Subtracting of two signed, decimal values.

DESCRIPTION: The content of REGB is subtracted decimally from the content of the FLAC, leaving the result in the FLAC. REGB and STO0 are not altered.

3.5.3 Multiply FLAC and REGB Decimally, Result to FLAC

MNEMONIC: MPY OPCODE: BD

STATUS:
If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION:
FLAC x REGB → FLAC
0 → STO0

PARAMETER: none

PURPOSE: Multiplying of two signed, decimal values.

DESCRIPTION: The content of the REGB is multiplied with the content of the FLAC, leaving the result in the FLAC. Storage Register STO0 is cleared. REGB is not altered. The numbers are treated as decimal ones.

3.5.4 Divide FLAC by REGB Decimally, Result to FLAC

MNEMONIC: DIV OPCODE: BE

STATUS:
If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC : REGB → FLAC

Remainder → STO0

PARAMETER: none

PURPOSE: Dividing of two signed, decimal values.

DESCRIPTION: The content of the FLAC is divided by the content of REGB. The result is left in the FLAC, the remainder in the Storage Register STO0. The numbers are treated as decimal ones. REGB is not altered.

Note: The division routine will hang endlessly if REGB contain zero. Before each division the content of REGB have to be tested with the instruction TSTRB to assure that it is not zero.

3.5.5 Add REGB to FLAC Hexadecimally, Result to FLAC

MNEMONIC: ADDH OPCODE: 0E

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC + REGB → FLAC

PARAMETER: none

PURPOSE: Adding of two hexadecimal values.

DESCRIPTION: The content of REGB is added to the content of the FLAC, the result is stored in the FLAC. The contents of both registers are treated as hexadecimal numbers in two's complement format. REGB and STO0 are not altered.

3.5.6 Substract REGB from FLAC Hexadecimally, Result to FLAC

MNEMONIC: SUBH OPCODE: 1E

STATUS:
If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC - REGB → FLAC

PARAMETER: none

PURPOSE: Subtracting of two hexadecimal values.

DESCRIPTION: The content of REGB is subtracted from the content of the FLAC, leaving the result in the FLAC. The contents of both registers are treated as hexadecimal numbers in two's complement format. REGB and STO0 are not altered.

3.5.7 Hexadecimal to Decimal Conversion

MNEMONIC: HEXDEC OPCODE: BF

STATUS:
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC = 0
POS = 1 NEG = 0 ZERO = 1

ACTION:
FLAC hexadecimally → FLAC decimally
NNN → REGB
0 → STO0

PARAMETER: none

PURPOSE: Converting of the hexadecimal FLAC value to decimal format for computations.

DESCRIPTION: The hexadecimal value, stored in digits 10E4 to 10E0 of the FLAC, is converted to decimal format. The decimal result is stored in the FLAC. REGB is destroyed by the conversion. Storage Register STO0 is cleared.

3.5.8 Round FLAC n Times

MNEMONIC: ROUNDn OPCODE: E0 - E3

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC : 10E(n-1) → FLAC
FLAC : 10 → FLAC with rounding

PARAMETER: None. The implicit operand n ranges from 1 to 4.

PURPOSE: Adjusting the FLAC when too large e.g. after a multiplication.

DESCRIPTION: The FLAC is shifted right n times. During the last shift the FLAC is rounded up or down depending on the value of the out shifted digit:

Digit < 5: no change of the new LSD
Digit > 4: the new FLAC is rounded up by 1

EXAMPLES: The FLAC is shown before and after rounding. Different contents and roundings are used:

FLAC before	FLAC after	Instruction
12345	1235	ROUND1
12345	123	ROUND2
98765432	98765	ROUND3
98765432	9877	ROUND4
0	0	ROUNDn
-15	-2	ROUND1

Example 4: FLAC before and after Rounding

3.5.9 Shift Right FLAC n Times

MNEMONIC: SHIFTRn OPCODE: E4 - E5

STATUS: If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC : 10En → FLAC (0 < n < 3)

PARAMETER: None. The implicit operand n ranges from 1 to 2.

PURPOSE: Adjusting of the FLAC.

DESCRIPTION: The FLAC is shiftet right n times. No rounding is made.

EXAMPLE: The FLAC's two LSDs are to be cleared.

```
...                  FLAC:            XXXX.YYYY
SHIFTR2              TRUNCATE LSDs      XXXX.YY
SHIFTL2              LSDs <- 00       XXXX.YY00
....                 NEXT INSTRUCTION HERE
```

Example 5: Adjusting of the FLAC by shifting

3.5.10 Shift Left FLAC n Times

MNEMONIC: SHIFTLn OPCODE: E6 - E7

STATUS:
If FLAC = 0:
POS = 1 NEG = 0 ZERO = 1
If FLAC > 0
POS = 1 NEG = 0 ZERO = 0
If FLAC < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC $\cdot$ 10En $\rightarrow$ FLAC $(0 < n < 3)$

PARAMETER: none

PURPOSE: Shifting left the FLAC n-times.

DESCRIPTION: The FLAC is shifted left one or two times. Zeroes are filled into the vacated nibbles of the FLAC.

EXAMPLE: The FLAC is to be shifted left twice freeing the 2 LSDs. These 2 digits are to be replaced by the value in REGB.

```
SHIFTL2          FLAC x 100       XXXXX00
ADD              REGB: YY         XXXXXYY
...
```

Example 6: Inserting to the FLAC by shifting left

3.6 Arithmetic Compare Instructions

3.6.1 Compare FLAC and REGB

MNEMONIC: CMPFLRB OPCODE: EC

STATUS:
If FLAC = REGB
POS = 0 NEG = 0 ZERO = 1
If FLAC > REGB
POS = 1 NEG = 0 ZERO = 0
If FLAC < REGB
POS = 0 NEG = 1 ZERO = 0

ACTION: FLAC - REGB → computed but not stored

PARAMETER: none

PURPOSE: Comparing the two signed values of FLAC and REGB.

DESCRIPTION: The contents of REGB are compared to the contents of the FLAC. This is done similiarly to a subtraction FLAC - REGB with the result stored nowhere. The Status Bits are set according to the following table:

SIGN	FLAC		+	+	-	-
SIGN	REGB		+	-	+	-
|FLAC|	>	|REGB|	POS	POS	NEG	NEG
|FLAC|	=	|REGB|	ZERO	POS	NEG	ZERO
|FLAC|	<	|REGB|	NEG	POS	NEG	POS

Figure 26: Flag Bit setting by the CMPFLRB instruction

3.6.2 Test Contents of REGB

MNEMONIC: TSTRB OPCODE: EE

STATUS: If REGB = 0
POS = 1 NEG = 0 ZERO = 1
If REGB > 0
POS = 1 NEG = 0 ZERO = 0
If REGB < 0
POS = 0 NEG = 1 ZERO = 0

ACTION: REGB compared to 0

PARAMETER: none

PURPOSE: Getting information concerning REGB.

DESCRIPTION: The content of REGB is compared to 0. The result is written to the Status Bits.

EXAMPLE: The content of REGB is tested before a division. If it contains 0. The division is not made due to the hangup caused by this REGB value.

```
MOVSTORB  2           MOVE STO2 -> REGB
TSTRB                 IF REGB = 0: ERROR OCCURED
JZ        DIVERR      JUMP TO ERROR HANDLING
DIV                   REGB # 0: FLAC : REGB -> FLAC
...
```

Example 7: Zero test before division

3.7 Bit Manipulation Instructions

3.7.1 Set Flag Bit

MNEMONIC: SBIT n OPCODE: 20 - 2F

STATUS: not affected

ACTION: 1 → addressed Flag Bit (0 ≤ n < 16)

PURPOSE: Setting a Flag Bit without affecting the other bits in the flag area.

DESCRIPTION: One of the 32 Flag Bits, selected by the operand n and the used Flag Group, is set to one in the flag area. The last SELGRPn instruction selects the used group.

PARAMETER: none

3.7.2 Reset Flag Bit

MNEMONIC: RBIT n OPCODE: 30 - 3F

STATUS: not affected

ACTION: 0 → addressed Flag Bit (0 ≤ n < 16)

PURPOSE: Resetting a Flag Bit without affecting the other bits in the flag area.

DESCRIPTION: One of the 32 Flag Bits, selected by the operand n of the instruction and the used Flag Group, is reset to zero in the flag area. The last SELGRPn instruction selects the used group.

PARAMETER: none

3.7.3 Test Flag Bit

MNEMONIC: TBIT n OPCODE: 40 - 4F

STATUS: If Bit = 0
ZERO = 1 NEG = 1 POS = 0
If Bit = 1
ZERO = 0 NEG = 0 POS = 1

ACTION: $\overline{\text{Addressed bit}} \rightarrow$ ZERO and NEG ($0 \le n < 16$)
Addressed bit $\rightarrow$ POS

PURPOSE: To test a selected Flag Bit if set or not.

DESCRIPTION: The addressed Flag Bit (by Group and bit number n) is transferred to the POS Bit. The inverted bit is transferred to the ZERO and NEG Bits. The last SELGRPn instruction selects the used group.

PARAMETER: none

3.7.4 Select Flag Bits Group n

MNEMONIC: SELGRPn OPCODE: F3 - F4

STATUS: not affected

ACTION: if n = 1: Select Group 1 flags
if n = 2: Select Group 2 flags

PURPOSE: Select the flags to be used.

DESCRIPTION: The selected group of flags is addressed for all bit modifying and testing instructions until an other SELGRPn instruction is encountered.

PARAMETER: none

3.7.5 OR FLAC and REGB

MNEMONIC: OR OPCODE: 0F

STATUS: If FLAC = 0
ZERO = 1 NEG = 0 POS = 1
If FLAC > 0
ZERO = 0 NEG = 0 POS = 1

ACTION: FLAC .OR. REGB → FLAC

PURPOSE: Oring the two register values.

DESCRIPTION: The logical OR function is applied to FLAC and REGB. Bits set in one or both registers are set in the FLAC. Bits not set in both registers are cleared in the FLAC.

PARAMETER: none

3.7.6 And FLAC and REGB

MNEMONIC: AND OPCODE: 1F

STATUS: If FLAC = 0
ZERO = 1 NEG = 0 POS = 1
If FLAC > 0
ZERO = 0 NEG = 0 POS = 1

ACTION: FLAC .AND. REGB → FLAC

PURPOSE: Anding the two register values.

DESCRIPTION: The logical AND function is applied to FLAC and REGB. Bits set in both registers are set in the FLAC, all others are cleared to zeroes in the FLAC.

PARAMETER: none

3.8 Constant Transfer Instructions

3.8.1 Load FLAC with a Positive Constant

MNEMONIC: LDFLPOS n OPCODE: 50 - 53

```
LDFLPOS n
BYTE     >NN
...
BYTE     >NN
```

STATUS: not affected

ACTION: + CONSTANT → FLAC ($0 \leq n < 4$)

PURPOSE: Loading the FLAC with a positive constant.

DESCRIPTION: The constant, which length is defined by the operand n, is loaded into the cleared FLAC with a positive sign. The constant follows the instruction in subsequent bytes. To assure the BCD format, the hex sign ">" has to be used for the constants.

PARAMETER: Defined by the operand n; n bytes follow. If n = 0; 4 bytes follow. Each byte contains two BCD numbers. The MSD of the operand is contained in the 1st byte after the instruction.

Note: It is the user's responsibility to append the correct number of bytes after the instruction. The assembler doesn't give a warning if the operand n and the number of appended bytes do not match. This is true for all four load instructions.

EXAMPLE: The FLAC is to be loaded with the constant +123456

```
LDFLPOS    3        3 BYTES WITH PARAMETERS FOLLOW
BYTE       >12      +123456 -> FLAC
BYTE       >34
BYTE       >56
ADD                 123456 + REGB -> FLAC
...                 NEXT INSTRUCTION HERE
```

Example 8: Loading the FLAC with a positive constant

3.8.2 Load FLAC with a Negative Constant

MNEMONIC: LDFLNEG n OPCODE: 54 - 57
BYTE >NN
...
BYTE >NN

STATUS: not affected

ACTION: - CONSTANT → FLAC

PURPOSE: Loading the FLAC with a negative constant.

DESCRIPTION: The constant, which length is defined by the operand n, is loaded into the cleared FLAC with a negative sign. The constant follows the instruction in subsequent bytes. To assure the BCD format, the hex sign ">" has to be used for the constants.

PARAMETER: Defined by the operand n; n bytes follow. If n = 0; 4 bytes follow. Each byte contains two BCD numbers. The MSD of the operand is contained in the 1st byte after the instruction.

3.8.3 Load REGB with a Positive Constant

MNEMONIC: LDRBPOS n OPCODE: 58 - 5B
BYTE >NN
...
BYTE >NN

STATUS: not affected

ACTION: + CONSTANT → REGB

PURPOSE: Loading REGB with a positive constant.

DESCRIPTION: The constant, which length is defined by the operand n, is loaded into the cleared REGB with a positive sign. The constant follows the instruction in subsequent bytes. To assure the BCD format, the hex sign ">" has to be used for the constants.

PARAMETER: Defined by the operand n; n bytes follow. If n = 0; 4 bytes follow. Each byte contains two BCD numbers. The MSD of the operand is contained in the 1st byte after the instruction.

3.8.4 Load REGB with a Negative Constant

MNEMONIC: OPCODE: 5C - 5F

```
LDRBNEG  n
BYTE     >NN
...
BYTE     >NN
```

STATUS: not affected

ACTION: - CONSTANT → REGB

PURPOSE: Loading REGB with a negative constant.

DESCRIPTION: The constant, which length is defined by the operand n, is loaded into the cleared REGB with a negative sign. The constant follows the instruction in subsequent bytes. To assure the BCD format, the hex sign ">" has to be used for the constants.

PARAMETER: Defined by the operand n; n bytes follow. If n = 0; 4 bytes follow. Each byte contains two BCD numbers. The MSD of the operand is contained in the 1st byte after the instruction.

EXAMPLE: REGB is to be loaded with the constant - 76543210.

```
LDRBNEG    0          4 BYTES WITH PARAMETERS FOLLOW
BYTE       >76
BYTE       >54        -76543210 ->REGB
BYTE       >32
BYTE       >10
ADD                   FLAC + (-76543210) -> FLAC
....                  NEXT INSTRUCTION HERE
```

Example 9: Loading of REGB with a negative constant

3.8.5 Clear FLAC Register

MNEMONIC: CLRFL OPCODE: D5

STATUS: not affected

ACTION: + 0 → FLAC

PURPOSE: Clearing of the FLAC.

DESCRIPTION: The constant +0 is loaded into the FLAC.

PARAMETER: none

3.8.6 Clear REGB

MNEMONIC: CLRRB OPCODE: D6

STATUS: not affected

ACTION: + 0 → REGB

PURPOSE: Clearing of REGB.

DESCRIPTION: The constant +0 is loaded into the REGB.

PARAMETER: none

3.9 Input/Output Instructions

3.9.1 Set R Output

MNEMONIC: SETR n OPCODE: 01 - 06
08 - 0D

STATUS: not affected

ACTION: 1 → DLn (0 < n < 14, n # 7)

PURPOSE: To set a Digit Latch to a logical one.

DESCRIPTION: The operand n selects the proper Digit Latch. If the operand n is from 1 to 6 inclusive, outputs R1 to R6 are set. For values greater than 7 the Digit Latches DL8 to DL13 are set.

PARAMETER: none

Note: R0 and R7 cannot be addressed due to their usage by the control software.

3.9.2 Reset R Output

MNEMONIC: RSTR n OPCODE: 11 - 16
18 - 1D

STATUS: not affected

ACTION: 0 → DLn (0 < n < 14, n # 7)

PURPOSE: To reset a Digit Latch to a logical zero.

DESCRIPTION: The operand n selects the proper Digit Latch. If the operand n is from 1 to 6 inclusive, outputs R1 to R6 are reset. For values greater than 7 the Digit Latches DL8 to DL13 are reset.

PARAMETER: none

Note: R0 and R7 cannot be addressed due to their usage by the control software.

3.9.3 Test Keyboard

MNEMONIC: TSTKEY >NN OPCODE: ED

STATUS: See Figure 27.

ACTION: Key status to Status Bits.
DL8 = 0

PURPOSE: Testing of a key connected to the K-Port or testing of a K-Port level.

DESCRIPTION: The addressed K-Port Kn is read in. This can be done with a set, reset or unchanged R-output. The input is compared with the information read in the last time and the Status Bits set according to the following list:

Transition	ZERO	POS	NEG
LO-LO	1	0	0
LO-HI	0	1	0
HI-LO	1	0	1
HI-HI	0	0	0

Figure 27: Status Bit setting by the TSTKEY instruction
The ZERO Bit shows the actual state of the key, the POS and NEG Bits show if transitions occured. The operand format is shown below:

0	R	STR	Kn

Figure 28: Operand byte of the TSTKEY instruction

R: Strobing R-output	00	R1
	01	R2
	10	R3
	11	R4
STR: Strobe used	x0	R-output not changed
	01	Negative strobe
	11	Positive strobe
Kn: K-input read in	00	K1
	01	K2
	10	K4
	11	K8

The read information without affecting the other K-inputs is stored and the addressed R-output set after the instruction according to the used strobe STR:
STR = x0: the R-output is not changed at all
STR = 01: the R-output is set to 1
STR = 11: the R-output is reset to 0

PARAMETER: 1 byte containing the operand. Appended by the assembler. See Figure 28 for details.

EXAMPLE: If the key, connected to R3 and K2 is pressed (LO-HI) the program has to continue at label PRESSK2. If the same key is released, the program has to continue at label RELEASK2

```
* TEST IF KEY R3/K2 CHANGED SINCE THE LAST READIN
*
          TSTKEY    >20+>0C+1 R3, POS. STROBE, K2
          JP        PRESSK2   KEY WAS PRESSED
          JN        RELEASK2  KEY WAS RELEASED
          ...                 NO CHANGE AT THE KEY
```

Example 10: Keyboard test example

3.9.4 Actualize K-Input and Timers

MNEMONIC: KINTIM OPCODE: EF

STATUS:
If K8 = K4 = K2 = K1 = 0
ZERO = 1 POS = X NEG = X
Otherwise
ZERO = 0 POS = X NEG = X

ACTION:
K8 → Group 1 Flag 15
K4 → Flag 14
K2 → Flag 13
K1 → Flag 12
Short Timer - Time difference → Short Timer
Long Timer - Time difference → Long Timer
If Short Timer ≤ 0
Short Timer + Short Timer Buffer → Short Timer
ST = 1 (Group 1 Flag 11)
If Long Timer ≤ 0
Long Timer + Long Timer Buffer → Long Timer
LT = 1 (Group 1 Flag 10)

PURPOSE: Updating timers and K-inputs during Active Mode.

DESCRIPTION: The Short Timer and Long Timer are actualized and the K-Port is read in. The actualization is only made for an activated timer, a stopped timer is not affected. The time difference since the last update is subtracted from the timers. If a timer reaches 0 or is counted below 0, the last loaded value, which is stored in the Timer Buffer, is added to the timer. This ensures stable timing as long as underflow doesn't occure twice in sequence. The appropriate flag is set, too. (The two flags ST and LT are reset only by the user's program, never by the interpreter). DL8 is not changed, if it is set, the contents of the K-Port Latch are read into the Group 1 flags 12 to 15.

PARAMETER: none

3.9.5 Read K-Input

MNEMONIC: KIN OPCODE: F2

STATUS:
If K8 = K4 = K2 = K1 = 0
ZERO = 1 POS = X NEG = X
Otherwise
ZERO = 0 POS = X NEG = X

ACTION:

0	→	DL8	
K8	→	Group 1	Flag 15
K4	→		Flag 14
K2	→		Flag 13
K1	→		Flag 12
K-Port	→		FLAC LSD

PURPOSE: Reading K-Port to flags and FLAC for testing.

DESCRIPTION: DL8 is switched to 0to set the K-Port to input direction. The K-Port is read into the Group 1 flags 12 to 15 and the FLAC's LSD without disturbing the other digits.

PARAMETER: none

3.9.6 Output LSD of FLAC to K-Port

MNEMONIC: FLKOUT OPCODE: F0

STATUS: not affected

ACTION:
1 → DL8
FLAC LSD → K-Port

PURPOSE: Outputting FLAC's LSD to the K-Port.

DESCRIPTION: The LSD of the FLAC is transferred to the K-Port Latch. DL8 is set, switching the K-Port to the output direction.

PARAMETER: none

EXAMPLE: The FLAC is to be output at the K-Port starting with the LSD. After each nibble R5 is to be set and reset for clocking.

```
R5          EQU         5
            ...
            LDCNT1      8           USE COUNTER 1 FOR COUNTING
LOOP        FLKOUT                  LSD OF FLAC -> K-PORT
            SETR        R5          OUTPUT CLOCK AT R5
            RSTR        R5
            SHIFTR1                 SHIFT NEXT NIBBLE TO LSD
            DECCNT1                 DECREMENT COUNTER 1
            JNZ         LOOP        IF COUNTER 1 # 0: NEXT NIBBLE
            ...                     IS 0: 8 NIBBLES ARE OUTPUT
```

Example 11: Outputting of the FLAC with clocks

3.9.7 Output Constant to K-Port

MNEMONIC: KOUT n OPCODE: F1

STATUS: not affected

ACTION: 1 → DL8
n → K-Port (0 ≤ n < 16)

PURPOSE: Outputting of constants to the K-Port.

DESCRIPTION: The operand n is transferred to the K-Port Latch. DL8 is set, switching the K-Port to the output direction.

PARAMETER: 1 byte containing the operand n. Two byte instruction.

EXAMPLE: K1 and K4 are to be set

```
            OUT         5           SET K1 AND K4
```

Example 12: Outputting of a constant to the K-Port

3.10 Rrogram Flow Control Instructions

See also chapter Subroutine Software.

3.10.1 Jump Unconditional

MNEMONIC: JMP LABEL OPCODE: 60 - 67

STATUS: not affected

ACTION: W → PC

PURPOSE: Allows the program to alter the normal sequential program execution.

DESCRIPTION: The JMP instruction jumps unconditionally to the address W specified by the label. The three MSBs are defined with 3 bits in the 1st byte of the instruction, the eight LSBs are defined by the 2nd byte of the instruction. The assembler generates both bytes of this 2 byte instruction automatically.

PARAMETER: None. Two byte instruction.

3.10.2 Jump if Zero, Jump if Equal

MNEMONICS: JZ LABEL OPCODE: 70 - 77
JEQ LABEL

STATUS: not affected

ACTION: If ZERO = 0 PC+2 → PC
If ZERO = 1 W → PC

PURPOSE: Allows the program to alter the normal sequential program execution if the ZERO Bit is set.

DESCRIPTION: The JZ or JEQ instruction jumps to the address W specified by the label if the ZERO Bit is set. If the ZERO Bit is reset, the instruction following will be executed.

PARAMETER: None. Two byte instruction.

3.10.3 Jump if not Zero, Jump if not Equal

MNEMONIC: JNZ LABEL
JNE LABEL — OPCODE: 78 - 7F

STATUS: not affected

ACTION:
If ZERO = 0 W → PC
If ZERO = 1 PC+2 → PC

PURPOSE: Allows the program to alter the normal sequential program execution if the ZERO Bit is reset.

DESCRIPTION: The JNZ or JNE instruction jumps to the address W specified by the label if the ZERO Bit is 0. If the ZERO Bit is set, the instruction following will be executed.

PARAMETER: None. Two byte instruction.

3.10.4 Jump if Positive

MNEMONIC: JP LABEL — OPCODE: 80 - 87

STATUS: not affected

ACTION:
If POS = 0 PC+2 → PC
If POS = 1 W → PC

PURPOSE: Allows the program to alter the normal sequential program execution if the POS Bit is set.

DESCRIPTION: The JP instruction jumps to the address W specified by the label if the POS Bit is 1. If the POS Bit is 0 the instruction following will be executed.

PARAMETER: None. Two byte instruction.

3.10.5 Jump if Negative

MNEMONIC: JN LABEL OPCODE: 88 - 8F

STATUS: not affected

ACTION: If NEG = 0 PC+2 → PC
If NEG = 1 W → PC

PURPOSE: Allows the program to alter the normal sequential program execution if the NEG Bit is set.

DESCRIPTION: The JN instruction jumps to the address W specified by the label if the NEG Bit is 1. If the NEG Bit is 0, the instruction following will be executed.

PARAMETER: None. Two byte instruction.

3.10.6 Call Subroutine

MNEMONIC: CALL LABEL OPCODE: 68 - 6F

STATUS: not affected

ACTION: SR2 → SR3
SR1 → SR2
PC+2 → SR1
W → PC

PURPOSE: Allows the program the control transfer to a subroutine. Because the CALL instruction saves the return address, subroutines may be called from various locations in a program. The subroutine will return control back to the proper, saved address after the return instruction RETN.

DESCRIPTION: The CALL instruction invokes a subroutine starting at the label's address W and pushes the return address onto the subroutine stack.
The stack's depth is 3. This allows 3 levels of subroutine nesting. If a 4th level of subroutine nesting is entered, the return address stored in SR3 is lost.

PARAMETER: None. Two byte instruction.

3.10.7 Return from Subroutine

MNEMONIC: RETN OPCODE: D4

STATUS: not affected

ACTION: SR1 → PC
SR2 → SR1
SR3 → SR2

PURPOSE: Returning from a subroutine to the address following the CALL of the subroutine.

DESCRIPTION: The Program Counter is loaded from the top of the subroutine stack. Then the stack is raised one level up, leaving the next return address on top of the subroutine stack. If the instruction RETN is used outside of a subroutine non predictable results will occur.

PARAMETER: none.

3.11 Analog-Digital-Converter Instructions

3.11.1 Set Converter Supply Voltage

MNEMONIC: SVDDON OPCODE: 07

STATUS: not affected

ACTION: SV_{DD} is switched on.

PURPOSE: Supplying the ADC part and sensors with voltage.

DESCRIPTION: The supply converter voltage is switched to the SV_{DD} pin and the internal resistor network of the ADC comparator.

PARAMETER: none

3.11.2 Reset Converter Supply Voltage

MNEMONIC:	SVDDOFF	OPCODE: 17
STATUS:	not affected	
ACTION:	SV_{DD} is switched off.	
PURPOSE:	Switching off the ADC part.	
DESCRIPTION:	The supply converter voltage SV_{DD} is switched off.	
PARAMETER:	none	

Note: The supply converter voltage SV_{DD} is also switched off too by initialization and the instructions DONE and OFF. Also the instruction MEASR switches the SV_{DD} off after the completed measurement.

3.11.3 Measure Addressed A/D Input

MNEMONIC: MEASR a OPCODE: AC - AF
BYTE NNN

STATUS:
If 0 < ADC < > FFF
POS = 1 ZERO = 0 NEG = 0

If ADC Value = 000
POS = 0 ZERO = 1 NEG = 1

If ADC Value → FFF
POS = 0 ZERO = 0 NEG = 1

ACTION:
DL9 ← 0
SV_{DD} on

Adding up ADC results of analog input A(a+1) in the ADC buffer. Final result → FLAC.
SV_{DD} off

PURPOSE: Single or double (compensated) measurements may be defined for the addressed analog input A(a+1).

DESCRIPTION: The analog input addressed by the operand a is switched to the ADC and measured CNT+1 times. CNT, which ranges from 0 to 127, is defined in the byte following the instruction. See Figure 29.

Compensated measurements: If the MSB M of the parameter byte is zero, CNT+1 measurements with noninverted comparator inputs and CNT+1 measurements with inverted comparator inputs are accumulated. This compensates the offset error of the internal comparator.
Single measurements: If the MSB M of the parameter byte is 1, CNT+1 measurements with noninverted comparator inputs are accumulated.

The ADC value is tested after each measurement: If an error occured, the NEG Bit is set to one and the measurements are terminated. The ZERO Bit shows if overflow or underflow occured. SV_{DD} is switched on before the measurements and is switched off after completion.

The result of the measurements is placed into the FLAC.

PARAMETER: The byte following the opcode contains the number of measurements CNT and the single/compensated information M in the MSB.

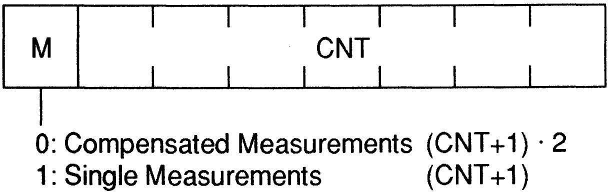

M
CNT
0: Compensated Measurements (CNT+1) · 2
1: Single Measurements (CNT+1)

Figure 29: Parameter byte of the MEASR instruction

EXAMPLE: Analog input A3 is to be measured 50 times in compensated mode. The result in BCD is to be loaded into the FLAC. If an error occurs program has to continue at labels OVFL resp. UNFL depending on the reason for the error.

```
A3        EQU       2
          ...
          MEASR     A3        MEASURE A3
          BYTE      49        ADD 49+1 COMPENSATED RESULTS
          JN        ERROR     NEG = 1 ? IF YES ERROR
          HEXDEC              NO, CONVERT RESULT TO BCD
          ...                 RESULT IN FLAC
ERROR     JZ        UNFL      UNDERFLOW OCCURED
OVFL      ...                 OVERFLOW OCCURED
```

Example 13: Measurement with error checking

3.11.4 Adjust Battery Voltage

MNEMONIC: ADJBATT OPCODE: E8

STATUS:
If 0 < ADC < > FFF
POS = 1 NEG = 0 ZERO = 0
If ADC = 000
POS = 0 NEG = 1 ZERO = 1
If ADC ≥ FFF
POS = 0 NEG = 1 ZERO = 0

ACTION:
1 → DL9
ADC → FLAC

PURPOSE: Battery measurement under Vddmin conditions for later battery checks.

DESCRIPTION: The value of the internal reference voltage is measured with a single measurement and the ADC hex value is stored in the 3 LSDs of the cleared FLAC. See examples in chapter 7.

PARAMETER: none

3.11.5 Check Battery Voltage

MNEMONIC: CHKBATT OPCODE: EA

STATUS: If Vdd > Vddmin:
POS = 1 NEG = 0 ZERO = 0
If Vdd < Vddmin
POS = 0 NEG = 1 ZERO = 0

ACTION: 1 → DL9
Comparison of the FLAC with the internal reference voltage.

PURPOSE: Battery voltage measurement for Vddmin check.

DESCRIPTION: The ADC measurement value of the internal reference voltage is compared against the hex value contained in the FLAC. Only a single measurement is made. See examples in chapter 7.

PARAMETER: none

3.11.6 Adjust Comparator Voltage

MNEMONIC: ADJCOMP OPCODE: E9

STATUS: If 0 < ADC < > FFF
POS = 1 NEG = 0 ZERO = 0
If ADC = 000
POS = 0 NEG = 1 ZERO = 1
If ADC ≥ FFF
POS = 0 NEG = 1 ZERO = 0

ACTION: 0 → DL9
ADC → FLAC

PURPOSE: ADC measurement for later comparisons.

DESCRIPTION: The ADC measurement value of the addressed analog input is measured with a single measurement. The ADC hex value is stored in the cleared FLAC's LSDs.

PARAMETER: none

Note: The analog input to be measured has to be addressed explicitly by DL10 and DL11. See Figure 6 for addressing.

3.11.7 Check Comparator Voltage

MNEMONIC: CHKCOMP OPCODE: EB

STATUS:
If Vadc > Vin:
POS = 1 NEG = 0 ZERO = 0
If Vadc < Vin:
POS = 0 NEG = 1 ZERO = 0

ACTION:
0 → DL9
Comparison of the FLAC with the addressed analog input.

PURPOSE: Comparison of input voltages with stored ADC values.

DESCRIPTION: The addressed analog input is compared against the hex value contained in the FLAC. Only a single measurement is made.

PARAMETER: none

Note: The analog input to be measured has to be addressed explicitly by DL10 and DL11. See Figure 6 for addressing.

EXAMPLE: The analog input A4 is to be measured for later comparisons. The ADC value is to be stored in STO4. The stored value is to be compared later against the analog input A3. If the voltage at A3 is higher than the stored value, then a subroutine LEVEL is to be invoked.

```
          SETR      DL10     ADDRESS A4: DL10 = DL11 = 1
          SETR      DL11
          ADJCOMP            MEASURE A4 AND STORE RESULT IN FLAC
          MOVFLSTO  4        STORE RESULT IN STO4
          ...
          ...
          SETR      DL11     ADDRESS A3 FOR COMPARISON
          RSTR      DL10     DL11 = 1, DL10 = 0, DL9 YET 0
          MOVSTOFL  4        LOAD STORED ADC VALUE
          CHKCOMP            COMPARE A3 WITH STORED ADC VALUE
          JP        NOSUB    POS = 1: ADC-VALUE > A3
          CALL      LEVEL    ADC-VALUE < A3: CALL SUBROUTINE
NOSUB     ...
```

Example 14: Comparing of two analog voltages

3.12 Display Instructions

The display used by the TSS400 Standard is configured like shown below:

	MSD						LSD
Digit n	1	2	3	4	5	6	7
Select	S1/2	S3/4	S5/6	S7/8	S9/10	S11/12	S13/14
Magnitude	10E6	10E5	10E4	10E3	10E2	10E1	10E0

Figure 30: Display configuration

3.12.1 Display Digit

MNEMONIC: DISPLDGn > NN OPCODE: B1 - B7

STATUS: not affected

ACTION: Byte info > NN → Digit n (0 < n < 8)

PURPOSE: Change of a single display digit e.g. after displaying the FLAC.

DESCRIPTION: The operand is interpreted in two ways as shown below. The resulting character is displayed in the addressed digit n.

O	H	Character Number					

1	H	Block		M	Z	O	

Figure 31: Operand byte of the DISPLDGn instruction

Two different formats exist for the operand. They differ in the MSB of the operand:

MSB = 0: The six least significant bits define the character number which is output to the addressed display digit. The character numbers are defined in Figure 15.

MSB = 1: The contents of the according FLAC digit (0-15) are output, depending on the bits H, M, Z and Block to the addressed display digit.

The bits H, Block, M and Z have the following meaning:

H: If set, the segment H of the digit is displayed independently of the according H-segment bit in Flag Group 1. If reset, the according H-segment bit in Flag Group 1 decides if the segment H is switched on or off.

Block: These two bits define the block of the character numbers to be used. If the contents of the current digit are to be output, the block 0 (numbers 0-9 and characters A-F) is to be used. Block 1 is to be used, if characters outside A - F are to be output. If combinations of the segments A F B

G E and H are to be output dependent on the contents of the current digit, the blocks 2 and 3 are to be used.

M: If set, a minus sign (segment G) is output at the addressed digit if the FLAC's sign is negative. If the FLAC's sign is positive or if M is reset, the addressed digit is handled normally.

Z: If set, leading zero suppression is made for the addressed digit. This means, the digit is blanked if it contains zero. If reset, a zero is treated like other numbers.

Note: The bits M and Z are normally senseless if not used with OPLA block 0. They should be zero if blocks 1 to 3 are used.

PARAMETER: 1 byte added by the asssembler containing the display information. Two byte instruction.

EXAMPLE: The FLAC is to be displayed with leading zeroes suppressed and no segment H. If Flag Bit 4 of Flag Group 2 is set, the 20th character of the OPLA is to be displayed at digit 1 with the segment H set. If the bit is reset, the 21st character without segment H is to be displayed.

```
H         EQU       >40
Z         EQU       >04
          ...
          DISPLFL   >17        DISPLAY FLAC DIGIT 1 - 7
          BYTE      20+H       DIGIT 1  ASSUME BIT 4 = 1
          BYTE      >80+Z      DIGIT 2  ZERO SUPPRESSED
          BYTE      >80+Z
          BYTE      >80+Z
          BYTE      >80+Z
          BYTE      >80+Z
          BYTE      >80        DIGIT 7  NO ZERO SUPPRESSION
          SELGRP2              TEST FLAG BIT 4 GROUP 2
          TBIT      4
          JNZ       BIT4SET    BIT 4 = 1: DISPLAY IS YET CORRECT
          DISPLDG1  21         BIT 4 = 0: DISPLAY TERM 21 (c)
BIT4SET   ...                  U.___123    IF FLAC = 123  OR
          ...                  c ___123    IF FLAC = 123
```

Example 15: Display of the FLAC with additional terms

3.12.2 Clear Display

MNEMONIC: DISPLCLR OPCODE: B8

STATUS: not affected

ACTION: Blanks → Display

PURPOSE: Clearing of the whole display.

DESCRIPTION: The display is filled with blank characters.

PARAMETER: none

EXAMPLE: The 22nd term of the OPLA is to be displayed at digit 2 with all other digits blanked:

```
DISPLCLR            BLANK THE WHOLE DISPLAY
DISPLDG2   22       DISPLAY TERM 22 AT DIGIT 2
...                 _h_____ -> DISPLAY
```

Example 16: Display of one term

3.12.3 Display FLAC

MNEMONIC: DISPLFL >NN OPCODE: B0
BYTE >MM
...
BYTE >MM

STATUS: not affected

ACTION: 1st byte info → Starting digit
2nd byte info → Starting digit + 1
...
Last byte info → Ending digit

PURPOSE: Display of the FLAC in parts or complete.

DESCRIPTION: The operand >NN defines the digits of the FLAC which are to be output to the display. The MSD defines the start digit, the LSD defines the end digit. E.g. >25 means, digit 2 to 5 (4 digits) are to be output. The display digits not involved in the operand stay unchanged. For each digit to be output, a parameter byte is needed. The number of parameter bytes is:

Operand's LSD - Operand's MSD + 1

The first parameter byte contains the information of the start digit, the last parameter byte contains the information of the end digit.

Note: It is the user's responsibilty to add the correct number of parameter bytes to the DISPLFL instruction. No errors or warnings are given by the assembler if the number mismatches with the instruction's operand.

The parameter bytes following the instruction are interpreted in two ways as shown below. The resulting character is displayed at the addressed digit:

O	H	Character Number

1	H	Block	M	Z	O

Figure 32: Parameter byte of the DISPLFL instruction

Two different formats exist for the parameters. They differ in the MSB of the parameters:

MSB = 0: The six least significant bits define the character number which is output to the according display digit. The character numbers are defined in Figure 15.

MSB = 1: The contents of the according FLAC digit (0-15) are output to the according display digit, depending on the bits H, M, Z and Block.

The bits H, Block, M and Z have the following meaning:

H: If set, the segment H of the digit is displayed independently of the according H-segment bit in Flag Group 1. If reset, the according H-segment bit in Flag Group 1 decides if the segment H is switched on or off.

Block: These two bits define the block of the character numbers to be used. If the contents of the current digit are to be output, block 0 (numbers 0 - 9 and characters A - F) has to be used. If characters outside A - F are to be output, Block 1 has to be used. If combinations of the segments A F B G E and H are to be output dependent on the contents of the current digit, block 2 and 3 have to be used.

M: If set, a minus sign (segment G) is output at the according digit if the FLAC's sign is negative. If the FLAC's sign is positive, a blank character is output. If reset, the according digit is handled normally.

Z: If set, leading zero suppression is made for the according digit. This means, the digit is blanked if it contains zero and all digits left to it, beginning at the first displayed digit in the FLAC, are zero, too. If reset, a zero is treated like other numbers.

Note: The bits M and Z are normally senseless if used with OPLA block 1 to 3. They should be zero if these blocks are used.

Figure 33 shows the displayed information depending on the variables H, M and Z (block 0). Display 1 shows a FLAC containing - 00001234, Display 2 shows it containing + 00000000. The instruction used for the figure follows. The term INFO contains the H, M and Z bit.

```
          DISPLFL    >17         DISPLAY FLAC DIGIT 1 TO 7
          BYTE       14          DIGIT 1 MSD: E
          BYTE       >80+Z
          BYTE       >80+INFO    DIGIT 3 CONTAINS THE VARIABLE
          BYTE       >80
          BYTE       >80
          BYTE       >80
          BYTE       >80         DIGIT 7 LSD
```

INFO:

H	M	Z	Display 1	Display 2
0	0	0	E_ 01234	E_ 00000
0	0	1	E__ 1234	E__ 0000
0	1	0	E_ - 1234	E__ 0000
0	1	1	E_ - 1234	E__ 0000
1	0	0	E_0.1234	E_0.0000
1	0	1	E__.1234	E__.0000
1	1	0	E_ -.1234	E__.0000
1	1	1	E_ -.1234	E__.0000

Figure 33: Display depending on the DISPLFL Para meter

PARAMETER: 1 byte added by the assembler containing the operand information >NN which digits are to be displayed. For each digit involved by the operand an additional parameter byte is to be added by the user.

EXAMPLE: The Storage Register 5 is to be displayed with the decimal point (segment H) on at digit 5. Leading zero suppression is needed. The sign is to be located at digit 1.

```
H         EQU        >40         H SEGMENT ON
M         EQU        >08         SIGN (-) ON IF NEGATIVE
Z         EQU        >04         LEAD. ZERO SUPPRESSION ON
          ...
          MOVSTOFL   5           STO5 -> FLAC
          DISPLFL    >17         DISPLAY FLAC: -___X.XX
          BYTE       >80+Z+M     DIGIT 1 CONTAINS SIGN
          BYTE      >80+Z     DIGIT 2
          BYTE      >80+Z     DIGIT 3
          BYTE      >80+Z     DIGIT 4
          BYTE      >80+H     DIGIT 5 WITH DEC. POINT
          BYTE      >80       DIGIT 6
          BYTE      >80       DIGIT 7
          ...
```

Example 17: Display with variables Z, M and H

EXAMPLE: The FLAC is to be output to the display digits 1 to 6. The digit 7 contains mode information in the especially designed segments A F B G and E. The mode information A F B G is stored in the FLAC's LSD. The segment E information is contained in bit 0 of Flag Group 1. The nonstructured combinations cannot occur.

```
BL2       EQU       >20       OPLA BLOCK 2 (A F B G)
BL3       EQU       >30       OPLA BLOCK 3 (A F B G E)
          ...
          DISPLFL   >16       DISPLAY FLAC DIGIT 1 TO 6
          BYTE      >80       DIGIT 1
          BYTE      >80       DIGIT 2
          BYTE      >80       DIGIT 3
          BYTE      >80       DIGIT 4
          BYTE      >80       DIGIT 5
          BYTE      >80       DIGIT 6
          SELGRP1             SEGMENT E ON ?
          TBIT      0
          JNZ       SEGEON    YES
          DISPLDG7  >80+BL2   NO, ONLY  A F B G
          ...
SEGEON    DISPLDG7  >80+BL3   YES, DISPLAY WITH E
          ...
```

Example 18: Display of segment combinations

3.13 Miscellaneous Instructions

3.13.1 Enter Done Mode

MNEMONIC: DONE OPCODE: D2

STATUS: not affected

ACTION: CPU part is switched off. SV_{DD} is switched off. EEPROM is switched off.

PURPOSE: Current saving by deactivation of unnecessary parts.

DESCRIPTION: The CPU part is switched off, mainly only RAM, the LCD driver and the timers stay active. The ADC supply voltage SV_{DD} and the EEPROM voltage are switched off, too. Wake-up occurs at the instruction following the DONE instruction. See Figure 8 and 9 for wake-up conditions.

PARAMETER: none

3.13.2 Enter Off Mode

MNEMONIC: OFF OPCODE: D3

STATUS: not affected

ACTION: CPU, SV_{DD}, EEPROM, LCD and timers are switched off.

PURPOSE: Current saving by deactivation of not necessary parts.

DESCRIPTION: The CPU and the timers are switched off, only the RAM stays active. Events at the inputs which would wake up the CPU from Done Mode, will wake up the CPU from Off Mode, too. See Figure 8 and 9 for the wake-up conditions. The ADC supply voltage SV_{DD} and the EEPROM voltage are switched off, too. Wake-up will occur at the address following the OFF instruction.

PARAMETER: none

3.13.3 No Operation

MNEMONIC: NOP OPCODE: 00 or FF

STATUS: not affected

ACTION: PC+1 → PC

PURPOSE: Code for doing nothing definitively: Deleting of opcodes during debugging. Insertion of delays.

DESCRIPTION: The next instruction is read.

PARAMETER: none

3.14 Counter Instructions

Two decimal counters exist (Counter 1 and Counter 2) which are identical in its single use. The descriptions use n for 1 or 2 respectively.

3.14.1 Decrement Counter n

MNEMONIC: DECCNTn OPCODE: D8 - D9

STATUS: If Result = 0
ZERO = 1 POS = X NEG = X
If Result # 0
ZERO = 0 POS = X NEG = X

ACTION: Counter n - 1 → Counter n (0 < n < 3)

PURPOSE: Counting down of Counter n decimally until zero is reached.

DESCRIPTION: The Counter n is decremented and the ZERO Bit set or reset according to the result.

PARAMETER: none

EXAMPLE: 50 compensated measurements are to be added for the ADC input A3. If an error occurs during these measurements, this is to be tried 5 times. After 5 effortless trials an error message is to be output.

```
A3          EQU         2
            ...
            LDCNT1      5           LOAD COUNTER 1 WITH 5 TRIALS
LOOP        MEASR       A3          ADD UP 50 COMP. MEASUREMENTS
            BYTE        49
            JN          CNTDWN      ERROR ?
            ...                     NO, MEASUREMENTS OK, PROCEED
CNTDWN          DECCNT1                 YES, DECR. TRIAL COUNT
            JNZ         LOOP        AND TRY ONCE MORE
            CALL        ERRMSG      5 EFFORTLESS TRIALS: ERROR
            ...
```

Example 19: Usage of a counter for repetitions

3.14.2 Increment Counter n

MNEMONIC: INCCNTn OPCODE: DB - DC

STATUS:
If Result = 0 (99 before)
ZERO = 1 POS = X NEG = X
If Result # 0
ZERO = 0 POS = X NEG = X

ACTION: Counter n + 1 → Counter n (0 < n < 3)

PURPOSE: Counting up of Counter n decimally until zero (100) is reached.

DESCRIPTION: The Counter n is incremented and the ZERO Bit set if the counter overflows to 100. The ZERO Bit is reset otherwise.

PARAMETER: none

3.14.3 Decrement Double Counter

MNEMONIC: DECDBL OPCODE: DA

STATUS: If Result = 0000
ZERO = 1 POS = X NEG = X
If Result # 0000
ZERO = 0 POS = X NEG = X

ACTION: A decimal counter consisting of Counter 2 (MSD) and Counter 1 (LSD) is decremented by 1.

PURPOSE: Counting down a decimal counter of doubled length.

DESCRIPTION: The counter consisting of the connected Counters 1 and 2 is decremented and the ZERO Bit set if the result is 0000. The ZERO Bit is reset otherwise.

PARAMETER: none

3.14.4 Increment Double Counter

MNEMONIC: INCDBL OPCODE: DD

STATUS: If Result = 0000 (9999 before)
ZERO = 1 POS = X NEG = X
If Result # 0000
ZERO = 0 POS = X NEG = X

ACTION: A decimal counter consisting of Counter 2 (MSD) and Counter 1 (LSD) is incremented by 1.

PURPOSE: Counting up a decimal counter of doubled length.

DESCRIPTION: The counter consisting of the connected Counters 1 and 2 is incremented and the ZERO Bit set if the counter overflows to 10000. The ZERO Bit is reset otherwise.

PARAMETER: none

3.14.5 Load Counter n Decimal

MNEMONIC: LDCNTn >NN OPCODE: DE - DF

STATUS: not affected

ACTION: >NN → Counter n (0 < n < 3)
PC+2 → PC

PURPOSE: Loading Counter n with a decimal value.

DESCRIPTION: The value of the operand is loaded into Counter n.The PC is adjusted to point to the next following instruction.

PARAMETER: 1 byte added by the assembler containing the decimal value in BCD format. ">" is used to assure BCD format. Two byte instruction.

EXAMPLES: The Counter 2 is to be loaded with the value 89:

```
LDCNT2    >89       LOAD COUNTER 2 WITH 89
...
```

The counters are to be loaded with the decimal value 1234 for combined counting.

```
LDCNT2    >12       LOAD COUNTER 2 WITH MSDs
LDCNT1    >34       LOAD COUNTER 1 WITH LSDs
...                 READY FOR DOUBLE INCR/DECR
```

Example 20: Loading of the counters

3.14.6 Move Counter n to REGB

MNEMONIC: MOVCNTn OPCODE: F5 - F6

STATUS: If Counter n = 00
ZERO = 1 POS = 1 NEG = 0
If Counter n # 00
ZERO = 0 POS = 1 NEG = 0

ACTION: 0 → REGB
Counter n → REGB (0 < n < 3)

PURPOSE: Loading Counter n to the REGB for comparing or computing.

DESCRIPTION: Counter n is transferred into the LSDs of the cleared REGB.

PARAMETER: none

3.14.7 Move Combined Counters to REGB

MNEMONIC: MOVDBL OPCODE: F7

STATUS: If Counters = 0000
ZERO = 1 POS = 1 NEG = 0
If Counters # 0000
ZERO = 0 POS = 1 NEG = 0

ACTION: 0 → REGB
Counter 1 → REGB 10E1 - 10E0
Counter 2 → REGB 10E3 - 10E2

PURPOSE: Loading of the combined counters to the REGB for comparing or computing.

DESCRIPTION: The combined counters are transferred into the LSDs of the cleared REGB. Counter 2 contains the MSDs.

PARAMETER: none

3.15 Timer Instructions

3.15.1 Load Timer Long

MNEMONIC: LDTIML NNN OPCODE: D0

STATUS: not affected

ACTION: NNN → Long Timer (0 ≤ NNN < 256)
NNN → Long Timer Buffer
0 → LT Group 1 Flag 10
start of Long Timer;

PURPOSE: Loading and starting of the Long Timer.

DESCRIPTION: The operand NNN is loaded into the timer counting seconds. A zero operand means 256 seconds. The value NNN is stored in the Long Timer Buffer too for restoring of the Long Timer when its contents reach or pass zero. The Group 1 Flag 10 (LT) is reset.

Note: The first timer interval after starting has an uncertain duration due to the asynchronous start inside the hardware timer's sequence. The first time interval has an uncertainity of up to 1 second.
The Long Timer remains active until the instruction STPTIML occurs.

PARAMETER: One byte added by the assembler containing the timer value. Two byte instruction.

EXAMPLE: The timer is to be loaded with 2 seconds.

```
LDTIML    2          LOAD SECONDS COUNTER
...                  WITH 2 SECONDS
```

Example 21: Loading of the long timer

3.15.2 Load Timer Short

MNEMONIC: LDTIMS NNN OPCODE: D1

STATUS: not affected

ACTION: NNN → Short Timer (0 ≤ NNN < 256)
NNN → Short Timer Buffer
0 → ST Group 1 Flag 11
start of Short Timer;

PURPOSE: Loading and starting of the Short Timer.

DESCRIPTION: The operand NNN is loaded into the timer counting 1/16 s. A zero operand means 256/16 seconds. The value NNN is stored in the Short Timer Buffer too for restoring of the Short Timer when its contents reach or pass zero. The Group 1 Flag 11 (ST) is reset.

Note: The first timer interval has an uncertain duration due to the asynchronous start inside the hardware timer's sequence. The first time interval has an uncertainity of up to 1/16 second.

The Short Timer remains active until the STPTIMS instruction occurs.

It is the user's responsibility to set the Digit Latch DL12 to one, if the Short Timer is to be used. The Short Timer will not operate with a reset DL12.

PARAMETER: One byte added by the assembler containing the timer value. Two byte instruction.

EXAMPLE: The Short Timer is to be loaded with 0.5 seconds. The Long Timer is to be loaded with 256 seconds.

```
SETR      DL12      SET DL12 FOR THE SHORT TIMER
LDTIMS    8         LOAD 1/16 S COUNTER: 0.5 S (8/16)
LDTIML    0         LOAD TIMER FOR SECONDS
...                 WITH 256 SECONDS (0)
```

Example 22: Loading of both timers

3.15.3 Actualize Timers

MNEMONIC: ACTTIM OPCODE: 10

STATUS: not affected

ACTION: Short Timer - Time difference → Short Timer
Long Timer - Time difference → Long Timer

If Short Timer ≤ 0
Short Timer + Short Timer Buffer → Short Timer
ST = 1 (Group 1 Flag 11)

If Long Timer ≤ 0
Long Timer + Long Timer Buffer → Long Timer
LT = 1 (Group 1 Flag 10)

PURPOSE: Updating Long and Short Timer during Active Mode.

DESCRIPTION: The Short Timer and Long Timer are actualized, if activated. A stopped timer is not affected by this instruction. The time difference since the last update is subtracted from the timers. If a timer reaches 0 or is counted below 0, the last loaded value, which is stored in the according Timer Buffer, is added to the timer. This ensures stable timing as long as an underflow doesn't occure twice in sequence. The appropriate flag is set, too. (The two flags ST and LT are reset only by the user's program, never by the interpreter). To ensure stable timing without loosing of information the updating has to be done in time intervals ti. The conditions for ti are:

ti < loaded time by LDTIMS or LDTIML
and ti ≤ 15/16 s for the Short Timer
ti ≤ 15 s for the Long Timer

PARAMETER: none

EXAMPLE: The timers are to be updated during a period of computation intensive tasks. The flag checking part is shown, too.

```
LT          EQU        10         LONG TIMER FLAG
ST          EQU        11         SHORT TIMER FLAG
            ...
            MOVSTORB    2         STO2 -> REGB
            MPY                   STO2 x FLAC
            ACTTIM                UPDATE TIMERS AFTER MULTIPLY
            SHIFTL2               FLAC x 100
            MOVPRMRB    CONSTANT  CONSTANT -> REGB
            DIV                   FLAC x 100 : CONSTANT
            ACTTIM                UPDATE TIMERS AFTER DIVISION
            SELGRP1               ADDRESS TIMER FLAGS
            TBIT       LT         LONG TIMER COUNTED DOWN ?
            JNZ        LTOV       YES
L$400       TBIT       ST         SHORT TIMER COUNTED DOWN ?
            JNZ        STOV
            ...                   CONTINUE PROGRAM
*
LTOV        RBIT       LT         RESET LT FLAG
            ...                   DO THINGS NECESSARY
            JMP        L$400      CHECK THE ST FLAG
STOV        RBIT       ST         RESET ST FLAG
            ...
```

Example 23: Updating of the timers during active mode

3.15.4 Stop Long Timer

MNEMONIC: STPTIML OPCODE: F8

STATUS: not affected

ACTION: The Long Timer is stopped.
0 → LT (Group 1 Flag 10)

PURPOSE: Deactivation of Long Timer.

DESCRIPTION: The Long Timer is stopped and the LT flag is reset. The LT flag stays reset until a LDTIML instruction restarts the Long Timer and underflow occurs afterwards.

PARAMETER: none

3.15.5 Stop Short Timer

MNEMONIC: STPTIMS OPCODE: F9

STATUS: not affected

ACTION: The Short Timer is stopped.

0 → ST (Group 1 Flag 11)

PURPOSE: Deactivation of Short Timer.

DESCRIPTION: The Short Timer is stopped and the ST flag is reset. The ST flag stays reset until a LDTIMS instruction restarts the Short Timer and underflow occurs afterwards.

PARAMETER: none

EXAMPLE: The Short Timer is to be loaded with 10 seconds. After this time a subroutine SUBR is to be called and the Short Timer deactivated.

```
          LDTIMS    160       10 SECONDS -> SHORT TIMER
          DONE
          SELGRP1
          TBIT      ST        SHORT TIMER REACHED 0 ?
          JNE       STRZ      YES
          ...                 NO, KEYBOARD ?
*
STRZ            STPTIMS             STOP SHORT TIMER
          CALL      SUBR      DO, WHAT IS TO BE DONE
          ...
```

Example 24: Loading and Stopping of a Timer

3.16 Host Control Instructions

These instructions are implemented mainly for the connection of the TSS400 Standard to a host (Slave Mode). They may be used in normal environments too, if this is valuable.

MNEMONIC: SLV >NN OPCODE: FA

STATUS: not affected

PURPOSE: I/O instructions when controlled by a host computer in the Slave Mode.

DESCRIPTION: The operand byte defines the function to be executed in the Slave Mode:

0	Code	x

Figure 34: Operand Byte of the SLV Instruction

Code 0 **Ouput Status Bits during Rx is HI**

The Sign nibble of the FLAC and the Status Bits are output to the I/O Pin in the following order: Sign, 3 dummy bits, POS, ZERO, NEG, 1 dummy bit. Rx is HI during the 8 bit transfer.

Code 1 **Output FLAC during Rx is HI**

The FLAC content is output to the I/O Pin starting with the MSB of the MSD and ending with the LSB of the Sign nibble. Then the Status Bits are output in the following order: POS, ZERO, NEG, 1 dummy bit. Rx is HI during the transfer of these 10 x 4 bits.

Code 2 **Read RAM in during Rx is HI**

The complete RAM is read in. The transfer starts with the MSB of the nibble Y = 0 located in X-bank 1 and ends with the LSB of the nibble Y = 15 in X-bank 15. See figure 37 for the sequence. Rx is HI during the transfer of these 9 x 16 x 4 bits.

Code 3 **Output RAM during Rx is HI**

The complete RAM is written out. The transfer starts with the MSB of the nibble Y = 0 located in X-bank 1 and ends with the LSB of the nibble Y = 15 in X-bank 15. See figure 37 for the sequence. Rx is HI during the transfer of these 9 x 16 x 4 bits.

Code 4 **Wait x seconds**

The program waits 0 to 15 seconds in Active Mode before the next instruction is read. The timing ambiguity is 1 second due to the random start.

Code 5 **Soft start**

The program starts at PC 000, the normal start point after initialization.

x: Codes 0-3: R-output Rx used for transfer indication The chosen R-output is set to HI during the transfer of the information. This allows distinction of the transferred information from the EEPROM control signals which use the I/O-Pin normally.

Code 4: waiting time in seconds

Code 5: not used

Note: The R-output used for transfer indication can range from R1 to R6. If no such signal is wished, DL10 or DL11 may be used. No checks are made to avoid the usage of R0, R7 or other digit latches.

More information concerning the above instructions is contained in chapter 9. See there for a description of the signals used for the transfers.

PARAMETER: 1 byte containing the actual I/O function and the R-Output for the clock resp. the waiting time.

4 Subroutine Software

4.1 General

By using the subroutine capability of the TSS400 Standard, programs are substantially compacted, enabling the user to write very powerful algorithms within the 512 (2048) word limit.

A subroutine is used to avoid duplication of EEPROM code when a particular section of code is used several times within a program. A subroutine is a section of code terminated with a RETN instruction. A CALL instruction transfers program execution to the first instruction in the subroutine. At the completion of the subroutine, program control is transferred to the instruction address immediately following the CALL instruction. Examples of a subroutine and different calling techniques will follow.

4.2 Nesting Subroutines

The TSS400 Standard has a subroutine stack with a depth of 3. This means that 3 levels of subroutines are possible (a subroutine can call a subroutine and this one can call a subroutine, too). This procedure is called nesting subroutines. If a 4th subroutine level is used, the return information of the 1st level is lost and the software will not work properly. So the user is advised always to be informed which subroutine level he is using when coding programs.

The same is true for the return instruction RETN. If this instruction is used when not inside a subroutine the Program Counter PC is filled with irrelevant data from the stack. Non predictable results will occur.

Example: The following program shows subroutine nesting 3 levels deep.

```
R5          EQU         5
            ...
MAINLOOP    CALL        TEST        INVOKE TEST SUBROUTINE
            ...                     FROM MAINLOOP: 1st LEVEL
*---------
TEST        CALL        PREPARE     CALL 2nd LEVEL
            CALL        DISPLAY     CALL 2nd LEVEL AGAIN
            RETN
*
```

```
PREPARE    CALL       CLRDISPL   CALL 3rd LEVEL
           SETR       R5         OUTPUT PULSE AT R5
           RSTR       R5
           ...
           RETN                  RETURN TO 1st LEVEL
*
CLRDISPL   DISPLCLR              ROUTINE CLEARS DISPLAY BUFFER
           ...
           RETN                  RETURN TO 2nd LEVEL
*
TEST       ...
           RETN
```

Example 25: Subroutine nesting levels

4.3 Multiple Entry Points

Often it is desired to use a subroutine several times, specifying different conditions each time for entering that subroutine. A call to the multiple entry points presets different conditions, and then a branch into the base subroutine is executed. Thus, rewriting the subroutine for each entry condition is avoided.

The following examples use the CREG routine as the basic subroutine.

```
CREG1      MOVSTORB   1          STO1 -> REGB
           JMP        CREG       USE CREG SUBROUTINE
*
CREG3      MOVSTORB   3          STO3 -> REGB. FALL INTO CREG
*
CREG       ADD                   FLAC + STOx -> FLAC
C1         MOVSTORB   2          STO2 -> REGB
           MPY                   RESULT x STO2 -> FLAC
           ROUND3                RESULT : 1000 -> FLAC
           RETN
```

Example 26: Multiple Subroutine entry points

Now calling sequences for computations can become:

```
CALL       CREG1     COMPUTATION WITH STO1
...
CALL       CREG3     COMPUTATION WITH STO3
...
MOVSTORB   4         COMPUTATION WITH STO4
CALL       CREG      USE BASIC ROUTINE
...
MOVSTORB   5         COMPUTATION WITH SUBTRACTED STO5
SUB
CALL       C1        USE ENTRY POINT C1
...
CALL       CREG      USE SUBROUTINE AS IT IS
...
```

Example 27: Subroutine calls

Note that the CREG subroutine is not modified and can be called again (like explained above with computation of STO4).

5 Correction of Tolerances by Software

The ADC conversion equations mentioned in 2.11 show nominal values which in reality differ due to tolerances of Vmin and Vmax. The sensors connected to the ADC have tolerances too, so calibration of the complete system is necessary by hardware (potentiometer, resistors) or by software.

Two possibilities exist for software calibration:

- Storage of the calibration factors in the TSS400 RAM
- Storage of the calibration factors in the EEPROM

The following chapters explain the two possibilities with examples.

5.1 Calibration Variables Stored in RAM

This kind of calibration variables storing is only possible if the TSS400 is connected always to a battery.

5.1.1 Computation with Nominal Values and Correction

Correction variables once written into the RAM via the I/O Pin are used after the A/D conversion and nominal value computation for correcting the result. This allows measurements with the noncalibrated device.

Example: RAM locations contain correction values C and D for adjusting of the temperature value to the corrected temperature T_{real}:

```
* Tnom  = N x 0.1363 - 6.696      nominal constants
* Treal = Tnom x C + D            Correction formula
* C = Slope correction  in STO2
* D = Offset correction in STO3
*
A1          EQU       0
            ...
            MEASR     A1        MEASURE A1
            BYTE      1         2 COMPENSATED MEASUREMENTS     XXXX
            JN        ERROR
            HEXDEC              CONVERT RESULT TO BCD          XXXX
            LDRBPOS   2         0.1363 -> REGB               0.YYYY
            BYTE      >13
            BYTE      >63
```

```
          MPY                                                  XXX.YYYY
          ROUND1                                               XXX.YYY
          LDRBPOS   2        6.696 -> REGB
          BYTE      >66
          BYTE      >96
          SUB
          ROUND1             TNOM -> FLAC                      XXX.YY
*
          MOVSTORB  2        C -> REGB                           X.YYY
          MPY                TNOM x C -> FLAC                   XX.YYYYY
          ROUND2             ROUND 2 DIGITS                     XX.YYY
          MOVSTORB  3        D -> REGB                          -X.YYY
          ADD                TNOM x C + D -> FLAC               XX.YYY
          MOVFLSTO  1        CORRECTED RESULT TO STO1
```

Example 28: Computation with nominal values and correction

5.1.2 Computation with Corrected Values only

Corrected variables once written into the RAM via the I/O Pin are used after the A/D conversion. The nominal constants have to be loaded for the 1st calibration measurements via the I/O or during the initialization by the user's program.

Example: RAM locations contain the corrected values C and D (slope and offset) for adjusting of the temperature value to the corrected temperature T_{real}:

```
 * Treal = N x C + D
 * C = Slope in STO2
 * D = Offset in STO3
 *
A1        EQU       0
          ...
          MEASR     A1       MEASURE A1
          BYTE      1        2 COMPENSATED MEASUREMENTS        XXXX
          JN        ERROR
          HEXDEC             CONVERT RESULT N TO BCD           XXXX
          MOVSTORB  2        C -> REGB                        0.YYYY
          MPY                N x C -> FLAC                   XX.YYYY
          ROUND1             ROUND 1 DIGIT                    XX.YYY
          MOVSTORB  3        D -> REGB                        -X.YYY
          ADD                N x C + D -> FLAC                XX.YYY
          MOVFLSTO  1        CORRECTED RESULT TO STO1
```

Example 29: Computation with corrected values only

5.2 Calibration Variables stored in EEPROM

If the correction variables are stored in the EEPROM the correction routine may look like the following example. This way is recommended if no battery is used. These EEPROM variables can be computed by a host computer or by the TSS400 Standard itself if a calibration software part is used. The nominal slope and offset values are part of the user's program. These values are overwritten with the corrected values after calibration. The example is the same as described before.

Example: EEPROM locations contain the corrected values C and D (slope and offset) for adjusting of the temperature value to the corrected temperature T_{real}:

```
* Tnom  = N x 0.1363 - 6.696      nominal constants
* Treal = N x C + D               Corrected formula
* C = Slope  in SLOPE
* D = Offset in OFFSET
*
A1        EQU       0
          ...
          MEASR     A1        MEASURE A1
          BYTE      1         2 COMPENSATED MEASUREMENTS    XXXX
          JN        ERROR
          HEXDEC              CONVERT RESULT TO BCD         XXXX
          MOVPRMRB  SLOPE     C -> REGB                   X.YYYY
          MPY                 N x C -> FLAC              XX.YYYY
          ROUND1              ROUND 1 DIGIT               XX.YYY
          MOVPRMRB  OFFSET    D -> REGB                   -X.YYY
          ADD                 N x C + D -> FLAC           XX.YYY
          MOVFLSTO  1         CORRECTED RESULT TO STO1
          ...
          ORG       500       START OF CONSTANTS IN EEPROM
SLOPE     BYTE      >00       AT ADDRESS 500
          BYTE      >00       +0.1363 INITIALLY
          BYTE      >13
          BYTE      >63
OFFSET    BYTE      >80       -6.696  INITIALLY
          BYTE      >00
          BYTE      >66
          BYTE      >96
          ...
```

Example 30: Calibration variables stored in the EEPROM

6 System Calibration

With a few simple instructions it is possible for the system to compute the calibration constants by itself.

Example: A thermometer at A1 is to be calibrated. Key 1 (R1/K1) is activated when the sensor is dived into a bath with 0 C. Key 2 (R1/K2) is activated when the sensor is dived into a bath with +80 C. When both measurements are made, the slope and offset are to be computed and stored in STO2 and STO1 respectively. The routine which computes the temperature T with these computed values is to be shown, too.

```
* T1 = ADC1 x SLOPE + OFFSET             0 C
* T2 = ADC2 x SLOPE + OFFSET            80 C
* --------------------------                         T2 - T1
* T2 - T1 = SLOPE (ADC2 - ADC1)   ->    SLOPE = -------------
*                                                   ADC2 - ADC1

*          80 - 0
* SLOPE = -----------
*         ADC2 - ADC1
*
* OFFSET = T1 - ADC1 x SLOPE       -> OFFSET = 0 - ADC1 x SLOPE
*
* THE FOLLOWING SUBROUTINE COMPUTES SLOPE AND OFFSET FOR
* THE TEMPERATURE SENSOR
*
A1         EQU        0          DEFINE A1 INPUT
           ...
CALIBRAT   TSTKEY     >0+>30+0   R1, POS. STROBE, K1
           JZ         CALIBRAT   KEY1 ACTIVE ?
           MEASR      A1         YES, MEASURE AT 0 C          XXXX
           BYTE       3          4 COMPENSATED MEAS. ADDED
           JN         ERROR      OUTSIDE RANGE
           HEXDEC                HEX -> BCD FLAC             XXXXX
           MOVFLSTO   1          SAVE ADC1 IN STO1
L$401      TSTKEY     >0+>30+1   R1, POS. STROBE, K2
           JZ         L$401      KEY2 ACTIVE ?
           MEASR      A1         YES, MEASURE AT 80 C         XXXX
           BYTE       3
           JN         ERROR      OUTSIDE RANGE
           HEXDEC                ADC2 -> FLAC                XXXXX
           MOVSTORB   1          ADC1 -> REGB                XXXXX
           SUB                   ADC2 - ADC1 -> FLAC          XXXX
           MOVFLRB               ADC2 - ADC1 -> REGB          XXXX
           LDFLPOS    0          +80.000000 -> FLAC             XX.YYYYYY
           BYTE       >80
```

```
          BYTE      >00
          BYTE      >00
          BYTE      >00
          DIV                 +80 : (ADC2 - ADC1) = SLOPE   0.00YYYY
          ROUND1              ROUND LAST DIGIT              0.00YYY
          MOVFLSTO  2         SLOPE -> STO2                 0.00YYY
          MOVSTORB  1         ADC1  -> REGB              XXXX
          MPY                 ADC1 x SLOPE = -OFFSET        0.YYYYY
          MOVFLSTO  1         -OFFSET -> STO1               0.YYYYY
          RETN
*
ERROR     ...                 ERROR HANDLING
RETN
```

Example 31: System calibration routine

```
* THE SUBROUTINE COMPUTES THE TEMPERATURE T OUT OF THE ADC VALUE
* AND THE STORED SLOPE (STO2) AND OFFSET (STO1). THE COMPUTED
* TEMPERATURE IS DISPLAYED
*
H         EQU       >40       H SEGMENT ON
Z         EQU       >04       LEAD. ZERO SUPPRESSION ON
M         EQU       >08       SIGN (-) ON IF NEGATIVE
*
COMPUTE   MEASR     A1        MEASURE A1
          BYTE      3         4 COMPENSATED MEAS. ADDED  XXXX
          JN        ERROR1
          HEXDEC              ADC VALUE -> FLAC IN BCD
          MOVSTORB  2         SLOPE -> REGB                 0.YYYYY
          MPY                 ADC x SLOPE -> FLAC           X.YYYYY
          MOVSTORB  1         -OFFSET -> REGB               0.YYYYY
          SUB                 ADC x SLOPE - (-OFFSET)      XX.YYYYY
          ROUND3              ADJUST TO XX.YY              XX.YY
*                             DISPLAY FLAC: LEAD. 0 SUPPR.
          DISPLFL   >17       DISPLAY FLAC: _-_XX.YY
          BYTE      16        DIGIT 1 ALWAYS BLANK
          BYTE      >80+Z+M   DIGIT 2 CONTAINS SIGN
          BYTE      >80+Z     DIGIT 3
          BYTE      >80+Z     DIGIT 4
          BYTE      >80+H     DIGIT 5 WITH DEC. POINT
          BYTE      >80       DIGIT 6
          BYTE      >80       DIGIT 7
          RETN
*
ERROR1    ...                 ERROR HANDLING
```

Example 32: Computation using system calibration variables

If the computed values for slope and offset are to be stored in the EEPROM, then the following exchanges have to be made:

MOVFLSTO	1	→	MOVFLPRM	OFFSET
MOVFLSTO	2	→	MOVFLPRM	SLOPE
MOVSTORB	1	→	MOVPRMRB	OFFSET
MOVSTORB	2	→	MOVPRMRB	SLOPE

7 Hints, Recommendations and Examples

7.1 Local Assembly Labels

A good programming practice is the use of "Local Assembly Labels". This means that labels, which are referenced only inside the subroutine where they are defined, have names in the form L$xxx, where xxx is a unique, decimal number. Otherwise labels which are entry points should get labels which give a certain description e.g. DIV, ADD. This simplifies the distinction between outside referenced labels and local labels.

The software examples use the above mentioned nomenclature whenever possible.

7.2 Integer Math Package

The package consists of four subroutines for signed addition, subtraction, multiplication and division. The sign is located in the MSB of the sign nibble.

The Integer Math Package consists of the following subroutines:

- ADD: adds REGB to the FLAC
- SUB: subtracts REGB from the FLAC
- MPY: multiplies REGB and FLAC
- DIV: divides REGB into the FLAC

The result always appears in the FLAC register.

Three RAM registers are used by the package:

FLAC: holds 1st operand and result after operation
REGB: holds 2nd operand (not modified by operation)
STO0: Used as a help register for multiplication and division. It can be used for intermediate storage as long as no division or multiplication is performed.

The following figure shows the registers like defined in the used Integer Math Package:

FLAC REGB STO0	10E7	10E6	10E5	10E4	10E3	10E2	10E1	10E0	SIGN
	MSD							LSD	SIGN

Figure 35: Register configuration of the iInteger Math Package

Before calling one of the four subroutines, the registers FLAC and REGB have to be loaded with the numbers to be processed. The result of the operation is always left in the FLAC register.

The package is an Integer Package which means that the position of the decimal point has to be controlled by the user. To show where the decimal point is assumed, one should note the format of the number at the right margin of the source as it is done in the following examples:

XXX.YY two digits right of the thought decimal point
XXXXXX integer number without fraction

The digit count of "X" can be used to show the largest possible number of digits left to the decimal point.

The rules for the position of the decimal point are:

- Addition: Positions after the decimal point have to be equal. The position is the same for the result.
- Subtraction: Same as addition.
- Multiplication: Positions after the decimal point may be different. Add the positions for the result position.
- Division: Positions after the decimal point may be different. Subtract the REGB position from the FLAC position to get the result position.

FLAC		REGB	RESULT IN FLAC
XXX.YYY	+	X.YYY	XXX.YYY
XXX.Y	–	XX.Y	XXX.Y
XXX.YYY	·	XX.Y	XXXXX.YYYY
XX	·	XX	XXXX
XXX.YYYY	:	XX.YYY	X.Y
XX.YY	:	XX	X.YY

Example 33: Decimal point location after computations

If two numbers have to be divided and the result should have n digits after the decimal point, the FLAC has to be loaded with the number shifted to the left appropriately and zeroes filled into the lower digits. The same procedure can be used if a smaller number is to be divided by a larger one.

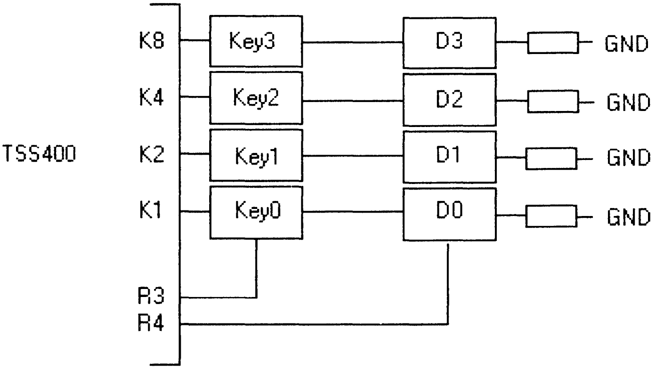
FLAC		REGB	RESULT IN FLAC
XXXX.000	:	XX	XX.YYY
XXXX.000	:	XX.Y	XX.YY
XXXX.000	:	X.YY	XXX.Y
0.YYY000	:	XX.Y	0.YYYYY

Example 34: Decimal point location for the division

It is the user's task to assure that no overflow conditions can occur. Before defining computations, a "worst case design" has to be made concerning the greatest numbers which have to be handled. If numbers grow too large, the previously described rounding routines (ROUNDn) may be used. If overflow occurs, no errors are reported! If the division is used it must be assured, that the divisor can't be zero: The software would stay endlessly in the division loop. This can be avoided by using the instruction TSTRB before any division.

7.3 Keyboard Scan

Up to 16 keys are possible with the four R-outputs and the four K-Ports. Static and dynamic states may be observed with the TSTKEY instruction. The following example shows an application with four keys and up to four diodes giving static information. The keys and diodes are connected to the TSS400 as follows (the pull-down resistors are shown too):



Key3: during activation no other key is to be recognized.
Key2: when activated (LO-HI) Counter 1 is to be incremented
Key1: when released (HI-LO) Counter 2 is to be displayed
Key0: when not activated Counter 1 is to be displayed
Diodes: are to be stored in Group 1 Flags 12 to 15

```
* KEYBOARD ROUTINE: 4 KEYS AND 4 DIODES ARE HANDLED
*
KEYBOARD   TSTKEY    >20+>C+3  R3, POS. STROBE, K8
           JNZ       CHKDIODE  IF HI (ZERO = 0) SKIP OTHER KEYS
           TSTKEY    >20+>C+2  R3, POS. STROBE, K4
           JP        KEY2LOHI  K2 LO-HI CHANGE ?
L$444      TSTKEY    >20+>C+1  R3, POS. STROBE, K2
           JN        KEY1HILO  K1 HI-LO CHANGE ?
L$445      TSTKEY    >20+>C+0  R3, POS. STROBE, K1
           JNZ       CHKDIODE
           MOVCNT1             NOT ACTIVATED, DISPLAY COUNTER 1
           EXCHRBFL            COUNTER 2 -> REGB -> FLAC
           DISPLCLR            CLEAR DISPLAY
           DISPLFL   >67       DISPLAY FLAC WITHOUT SEGMENT H
           BYTE      >80       DIGIT 6: 10E1
           BYTE      >80       DIGIT 7: 10E0
*
CHKDIODE   SETR      R4        READ DIODES INTO FLAGS 12-15
           KIN
```

```
          RSTR      R4
          RETN
*
KEY2LOHI  INCCNT1             K2 LO-HI CHANGE: INCR. COUNTER 1
          JMP       L$444
*
KEY1HILO  MOVCNT2             K1 HI-LO CHANGE: DISPLAY COUNTER 2
          EXCHRBFL            COUNTER 2 -> REGB -> FLAC
          DISPLCLR            CLEAR DISPLAY
          DISPLFL   >67       DISPLAY FLAC, NO SEGMENT H
          BYTE      >80       DIGIT 6: 10E1
          BYTE      >80       DIGIT 7: 10E2
          JMP       L$445
```

Example 35: Keyboard with 4 keys and 4 diodes

If more than 16 keys have to be handled, R5 and R6 can be used, too. They have to be set and reset by additional instructions placed before and after the TSTKEY-instruction. Additional the STR-bits of the TSTKEY-instruction should be set to 00 to ensure, that no other R-output is modified.

7.4 Clock Routine

The actual time is stored in Storage Register 1 in minutes. If the time is to be displayed, the time is computed into the format XX.YY (00.00 to 23.59) by a subroutine called CLOCK24. The same procedure may be used for inclusion of the day of the week.

```
LT        EQU       10        DEFINE LONG TIMER FLAG
H         EQU       >40       DECIMAL POINT SEGMENT H
          ...
          LDTIML    60        LOAD LONG TIMER WITH 60 SECONDS
IDLE      DONE                SLEEP UNTIL NEXT MINUTE IS OVER
          CALL      CLOCK24   CLOCK ROUTINE OUTPUTS TIME TO LCD
          ...                 OTHER ACTIVITIES, AFTER COMPLETION
          JMP       IDLE      JUMP BACK TO 1 MINUTE LOOP
*
* THE CLOCK ROUTINE COMPUTES HOUR AND MINUTES OUT OF THE
* ACCUMULATED MINUTES IN STO1.
*
CLOCK24   SELGRP1             SELECT GROUP 1 FLAGS
          TBIT      LT        LONG TIMER COUNT DOWN TO 0 ?
          JZ        NOACT     NO, WAKEUP NOT CAUSED BY TIMER
```

```
*
* THE LONG TIMER REACHED ZERO: INCREMENT MINUTES
*
          RBIT      LT        RESET UNDERFLOW FLAG LT
          MOVSTOFL  1         COUNTED MINUTES TO FLAC      XXXX
          LDRBPOS   1         ADD 1 MINUTE
          BYTE      >01
          ADD
*
* CHECK IF 24.00 IS REACHED: COUNTER CONTAINS 1440 OR MORE
*
          LDRBPOS   2         24.00 REACHED ?
          BYTE      >14       24 x 60 = 1440 MINUTES
          BYTE      >40
          CMPFLRB             FLAC - REGB: STATUS BITS SET
          JN        GOON      TIME < 24.00 (1440 MINUTES)
          SUB                 TIME >= 24.00: SUBTRACT 1440 MINUTES
GOON      MOVFLSTO  1         CORRECTED TIME BACK TO STO1
*
* THE MINUTES ARE DIVIDED BY 60. THE FLAC CONTAINS THE HOURS
* AFTERWARDS, STO0 THE MINUTES AS REMAINDER.
*
          LDRBPOS   1         60 -> REGB
          BYTE      >60
          DIV                 MINUTES : 60 -> FLAC (HOURS)
          SHIFTL2             HOURS x 100                  XX00
          MOVSTORB  0         MINUTES (REMAINDER STO0)       YY
          ADD                 ADD HOURS AND MINUTES        XXYY
*
* DISPLAY TIME: ZERO SUPPRESSED, DECIMAL POINT AT DIGIT 5
*
          DISPLCLR            CLEAR DISPLAY
          DISPLFL   >47       OUTPUT FLAC              ___XX.YY
          BYTE      >80       DIGIT 4: HOURS 10E1
          BYTE      >80+H     DIGIT 5: HOURS 10E0 + DEC. POINT
          BYTE      >80       DIGIT 6: MINUTES 10E1
          BYTE      >80       DIGIT 7: MINUTES 10E0
NOACT     RETN
```

Example 36: 24 Hour clock routine

7.5 Temperature Computation for Sensors

Shown is a subroutine which computes the nominal temperature out of the ADC value in the FLAC. The used TSS400 values are corresponding to Spec. Rev. 1.0. One compensated measurement is made.

```
* SUBROUTINE COMPUTES TEMPERATURE FROM A/DC VALUE OF ONE
* COMPENSATED MEASUREMENT.
*
* INPUT:         ADC VALUE    (>2 - >1FFC)  IN A/D BUFFER
* AFTER RETURN:  TEMPERATURE T (+-XXX.YY C)  IN FLAC
*
* BASE OF COMPUTATIONS:
*
* RV        = 5.11 KOHM                     SERIES RESISTOR
* R25       = 2.00 KOHM                     SENSOR RESISTOR @25C
* R85/R25   = 1.54                          SENSOR INCLINATION
* ADC VALUE 0002 EQUALS  0.101309 x SVDD    LOWER SPEC A/D VALUE
* ADC VALUE 1FFC EQUALS  0.494607 x SVDD    UPPER SPEC A/D VALUE
* FROM THE ABOVE:
* ADC VALUE = Vin/SVDD x 20813.7699 - 2106.6282 A/DC EQUATION
* T = ADC VALUE x 0.030416508 - 89.005919       EXACTLY
* T = ADC VALUE x 0.03042     - 89.01           USED
*
*  CALL:  ADC VALUE   (0 - >1FFC)    IN FLAC
* RESULT: TEMPERATURE (XXX.YY)       IN FLAC
*
TEMPCOMP  HEXDEC              BCD VALUE OF ADC VALUE   XXXX
          LDRBPOS   2         0.03042 -> REGB
          BYTE      >30                                   0.0YYYY
          BYTE      >42
          MPY                 ADC-VALUE x 0.01363       XXX.YYYYY
          ROUND3                                        XXX.YY
          LDRBNEG   2
          BYTE      >89       -89.01 -> REGB             XX.YY
          BYTE      >01
          ADD                 ADC-VALUE x 0.03042 - 89.01
          RETN                TEMPERATURE IN FLAC         X.YY
```

Example 37: Temperature computation routine

7.6 Battery Check

If regular battery checks are necessary, the ADC value of the internal reference voltage is to be measured with the critical low Vdd value. This ADC value is stored in the EEPROM and transferred to the FLAC when a battery check is necessary. The measured ADC value is compared to this stored value and the Status Bits set according to the result:

If Vdd > Vddmin: POS = 1 ZERO = 0 NEG = 0
If Vdd < Vddmin: POS = 0 ZERO = 0 NEG = 1

```
* Vddmin IS APPLIED TO Vdd AND A MEASUREMENT MADE TO GET
* THE WARNING VALUE FOR LATER BATTERY CHECKS
*
MEASVDDM  ADJBATT             MEASURE REFERENCE VOLTAGE
          MOVFLPRM  BATTMIN   WITH VDDMIN APPLIED
          ...                 RESULT IN FLAC TO EEPROM (HEX)

* TIME HAS COME FOR A BATTERY CHECK
*
          MOVPRMRB  BATTMIN   ADC VALUE FOR VDDMIN -> REGB
          MOVRBFL             REGB -> FLAC
          CHKBATT             COMPARE WITH ACTUAL VDD
          JN        BATTLO    NEG = 1: VDD < VDDMIN  WARNING !
          ...                 POS = 1: VDD IS OKAY
          ORG       508
BATTMIN   BYTE      0         STORAGE FOR ADC VALUE
          BYTE      0         OF VDDMIN
          BYTE      0         FORMAT: 00000XXX
          BYTE      0         LSD
```

Example 38: Vddmin setting and Battery Check

7.7 Cold Start, Warm Start

The TSS400 Standard checks after each initializaton if Cold Start or Warm Start occured. If the RAM locations "identity", loaded by the interpreter, contain >A5, the interpreter assumes Warm Start occured. If these locations contain other numbers, Cold Start is assumed and the RAM is cleared.

See 2.13 for description of Warm Start and Cold Start.

If it is necessary to continue at a certain program part after a Warm Start condition and not at the Cold Start address, one of the flags in the Flag Registers may be used for this purpose. This flag is set by the user's program to one after the Cold Start and may be tested when the program starts at PC 000:

If the flag is 0: Cold Start occured (RAM reset by interpreter)
If the flag is 1: Warm Start occured (RAM not modified)

The following example shows this approach to distinguish between Cold Start and Warm Start when the program has to continue at different program parts:

```
          ORG       000       START AFTER HARDWARE INIT
POWERUP   SELGRP1             USE GROUP1 FLAG 9
          TBIT      9
          JNZ       WARMSTRT  IF SET: WARMSTART OCCURED
*
* COLDSTART: BIT RESET BY INTERPRETER'S RAM CLEAR ROUTINE
*
          SBIT      9         WARM START BIT <- 1
          CALL      INIT      INITIALIZE SYSTEM
          ...
*
* WARMSTART: CONTINUATION AFTER EMI INDUCED INITS
*
WARMSTRT  ...                 RECOVER PART
```

Example 39: Cold Start/Warm Start distinction

7.8 Waiting Routines

It is often necessary to wait a certain or minimum time before the processing may continue. For this purpose the two timers may be used:

- The timer is loaded with the appropriate value.
- The Done Mode is activated with the instruction DONE.
- When the chosen time has elapsed the program will continue at the address following the DONE instruction. If changes at the K inputs can occur, a test has to be made if this was the reason for a wake-up.

Example: The FLAC is to be displayed for 2 seconds. Then the display is to be cleared. Changes at the K-Port may not shorten the waiting time. The Short Timer is to be deactivated after the elapsed delay.

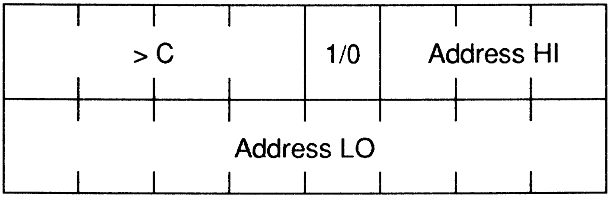
```
          DISPLFL   >17       DISPLAY FLAC 1 - 7
          BYTE      >80       DIGIT 1
          BYTE      >80
          BYTE      >80
          BYTE      >80
          BYTE      >80
          BYTE      >80
          BYTE      >80       DIGIT 7
          LDTIMS    32        LOAD 1/16 S TIMER WITH 32 x 1/16 S
WAIT2S    DONE                DONE MODE FOR 2 SECONDS
          SELGRP1             SELECT GROUP 1 FLAGS
          TBIT      ST        WAKEUP CAUSED BY SHORT TIMER ?
          JZ        KEY       NO, IF ST = 0
          STPTIMS             YES, CAUSED BY THE SHORT TIMER
          DISPLCLR            CLEAR THE DISPLAY, 2 S ELAPSED
          ...                 CONTINUE WITH PROGRAM
*
KEY       ...                 CAUSED KEYBOARD THE WAKEUP ?
JMP       WAIT2S    WAIT UNTIL 2 S ELAPSED
```

Example 40: Timer usage for waiting purposes

7.9 Address Modification

The EEPROM store and retrieve instructions MOVFLPRM and MOVPRMRB use fixed addresses inside the EEPROM. If tables are used, each item to be stored or retrieved needs an instruction which addresses it explicitly.

A way to circumvent this EEPROM consuming approach is to modify the address part of the MOVFLPRM or MOVPRMRB instructions:



The steps needed are:

- initializing of the instruction with the address of the 1st address (highest address in our example)
- storage or retrieving of an item
- transfer of the complete instruction to the FLAC
- subtraction of the item length from the instruction
- transfer of the modified instruction back to the original place
- check if the lowest address is reached and terminating resp. Looping back dependend on the result

Note: Starting with the lowest address and incrementing is possible too, but left shifts are needed if the EEPROM space is to be used efficiently. Backward stepping allows the adaptation to the actual length of information by overwriting of the leading zeroes of the item before.

Two NOPs are necessary before the MOVFLPRM or MOVPRMRB instructions due to the length of the FLAC, which is 4 bytes.

Example: Data with four digits (XX.YY) is to be stored downwards from EEPROM address >6FF to >600:

```
* INITIALIZATION OF THE DATA POINTER
*
LENGTH     EQU       2         ITEM LENGTH IN BYTES (4 DIGITS)
           ...
           LDFLPOS   2         2 BYTES
           BYTE      >C0+>6    OPCODE MOVFLPRM: >C0
           BYTE      >FF-3     1. BYTE OF 4 TO >6FC
           MOVFLPRM  POINTER   0000C6FC -> POINTER
           ...
*
* THE LOOP FOLLOWS:
* THE CONTENTS OF THE FLAC ARE STORED DOWNWARDS
*
POINTER    NOP                 THE FOLLOWING 4 BYTES ARE
           NOP                 MODIFIED AFTER EACH TRANSFER.
           MOVFLPRM  POINTER   THE MODIFIED INSTRUCTION
*
* DECREMENTING BY 2 FOR THE NEXT ITEM. THE LEADING NOPs
* ARE OVERWRITTEN.
*
           MOVPRMRB  POINTER   CURRENT POINTER -> FLAC
           MOVRBFL
           LDRBPOS   1         ITEM LENGTH IN BYTES
           BYTE      LENGTH    TO REGB
           SUBH                POINTER - LENGTH -> POINTER
           MOVFLPRM  POINTER   NEW POINTER BACK
*
* CHECK IF THE LOWEST ADDRESS IS REACHED
*
           LDRBPOS   2         LAST USEABLE ADDRESS
           BYTE      >C0+>6    >600 + OPCODE -> REGB
           BYTE      >00       >600 AND >601 ALWAYS 00
           CMPFLRB
           JP        POINTER   NOT YET REACHED
           ...                 ALL ITEMS STORED
           ...
           ORG       >600      START OF TABLES
           BYTE      0         ALWAYS 0
           BYTE      0         ALWAYS 0
BYTE       0        XX.    ITEM 127   >602
BYTE       0        YY     ITEM 127   >603
BYTE       0        XX.    ITEM 126   >604
BYTE       0        YY.    ITEM 126   >605
...
BYTE       0        XX.    ITEM 2     >6FC
BYTE       0        YY     ITEM 2     >6FD
BYTE       0        XX.    ITEM 1     >6FE
BYTE       0        YY     ITEM 1     >6FF
...
```

Example 41: Value storage in EEPROM

If data is to be retrieved from the EEPROM, the same way may be used as shown in the example above. The only difference is the opcode of the instruction:

```
MOVFLPRM   Opcode >C0
MOVPRMRB   Opcode >C8
```

Note: If more tables are used, it is adviseable to use more than one pointer. This reduces the necessary write cycles to the EEPROM bytes used as pointers. Another way would be the usage of PCF8570 RAMs for the pointer locations. See 2.16 for more information.

7.10 Square Root Routine

The routine shown computes the square root of the FLAC. The FLAC must contain a value A with an even number of digits after the imaginal decimal point. The result has the half number count of digits after the imaginal decimal point. The used approximation is:

Xn + 1 = (A/Xn + Xn)/2

The first part of the subroutine looks for a good starting value X1 (eg. 10 for A = 1 to 99, 100 for A = 100 to 9999) the second part starts with this value 10 approximations, leaving the result in the FLAC.

```
* SQUARE ROOT ROUTINE: THE SQUARE ROOT OF THE FLAC IS
* COMPUTED WITH THE NEWTONIAN PROCEDURE
* FLAC BEFORE: XXXXXX     XXXX.YY       XX.YYYY   NUMBER
* FLAC AFTER:     XXX       XX.Y          X.YY     ROOT
*
SQROOT     MOVFLSTO  2         STORE VALUE A IN FLAC
           LDRBPOS   1         FIND VALUE X1 FOR 1ST TRIAL
           BYTE      1
PLOOP      EXCHRBFL
           SHIFTL1
           EXCHRBFL
           SHIFTR2
           JNZ       PLOOP
           MOVRBSTO  1         STORE X1
           LDCNT1    10        10 TRIALS ARE MADE
LOOP       MOVSTOFL  2         A -> FLAC
           MOVSTORB  1         Xn -> REGB
           DIV                 A / Xn
           ADD                 (A / Xn) + Xn
           LDRBPOS   1
           BYTE      2
           DIV                 ((A / Xn) + Xn) / 2
           MOVFLSTO  1         Xn+1 -> Xn
           DECCNT1             ALL TRIALS MADE ?
           JNZ       LOOP
           RETN                YES, SQUARE ROOT IN FLAC
```

Example 42: Square Root routine

8 Development Tools

The keys to be pressed at the host computer are shown in brackets. The meaning is as follows:

<RETN> The key, here the return key, is to be pressed
<SHIFT/F3>The keys, here the shift and the F3 key, are to be pressed simultaneously

The cursor may be moved with the arrow keys, the acknowledge is made with the return key.

8.1 The Development Board

For the development of the users software including the debugging phase, a special hardware, the Development Board is necessary. This Development Board contains all necessary hardware needed for debugging and the final application:

- sockets for the TSS400 and the EEPROMs
- connectors to the users hardware (I/O and display)
- connector to the host computer
- interface to the host computer

If the ADT400 is used for development, the connector of the ADT400 is inserted into the microcomputer socket.

If no ADT400 is used, the TSS400 Standard is inserted into the microcomputer socket.

The schematic of the Development Board is shown in the following figure. If an X24C16 is used, it must be placed in socket 0.

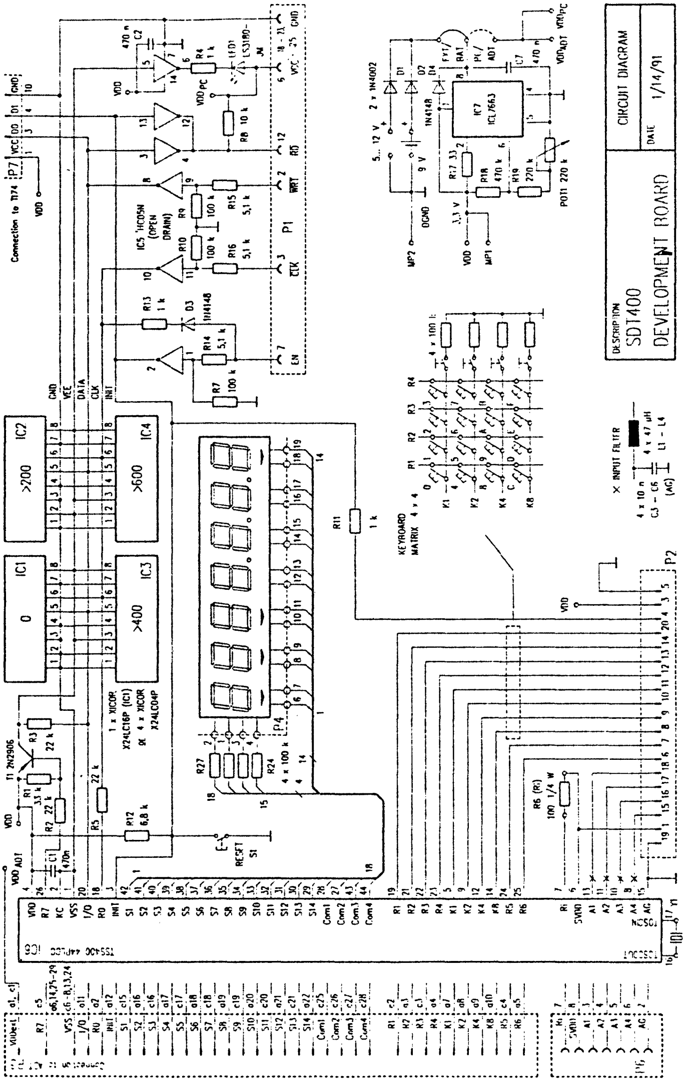


Figure 36: Schematic of the Development Board

8.2 Development with the ADT400

Necessary components for the software development are:

- IBM compatible Personal Computer
- ADT400 hardware connected to the Personal Computer
- ADT400 development software installed on the Personal Computer
- Development Board with EEPROM(s)

For details concerning the connection of the ADT400 to the host computer see the "TSS400 ADT USER'S GUIDE".

The TSS400 Standard software running on the ADT400 is delivered in a version which allows "worst case" real time tests. All necessary runtime definitions of the ADT400 are chosen to allow this. The used main oscillator frequency is 500 kHz. This is the minimum frequency of the TSS400.

When using the ADT400 the TSS400 Standard software is loaded into the ADT400. Then the ADT400 is used like the TSS400 Standard. The interpreter code is developed with the host computer's editor and assembled with the ASM400 assembler. The assembler outputs on request up to four TSS400 objects which contain the user's object together with an EEPROM burning software for transferring the object to the EEPROMs. After burning of the EEPROMs the TSS400 Standard software is loaded into the ADT400 system and the user's software may be debugged or run under control of the ADT400 emulator.

8.2.1 Assembling the User's Software

The ASM400 assembler for the TSS400 family includes the instruction set of the TSS400 Standard interpreter. To load the right instruction set, the 1st line of the user's code must look like the 1st line of the following example where xxxxxx is the title of the program. TSS444 or TSS0444 is the code which invokes the interpreter's instruction set. The 2nd line of code is necessary to have the arithmetic functions of the assembler available (+ - * : >).

```
TITLE    xxxxxx    TSS0444
OPTION   ARITHM
...
...
END
```

The users code must end with the "END" directive as shown above. Additional information concerning the ASM400 assembler may be found in the **"4-Bit Microcomputer Cross Assembler User's Guide"**.

The ASM400 is called by the following sequence:

```
 C:/ASM400                          <RETN>
INSTRUCTION SET FILE [INSTRUCT.400]: INSTRUCT.400   <RETN>
                        SOURCE FILE: xxxxxx.SRC     <RETN>
                        OBJECT FILE: xxxxxx.OBJ     <RETN>
                          OPLA FILE: xxxxxx.OPL     <RETN>
                       LISTING FILE: xxxxxx.LST     <RETN>
```

After the input of all filenames the assembler starts and will output the following status messages:

```
READ INSTRUCTION SET
CONTINUE WITH PASS1
CONTINUE WITH PASS2

0 ERROR(S)   0 WARNING(S)  IN MODULE xxxxxx
```

After completion of the assembly run, the ASM400 asks if the EEPROM burning files are wished. If they are wished they are created and written to the disk:

```
Transferroutine:        → BURN.BR#
```

Note: If a Burn-File is necessary and not created by the delivered ASM400 version, please contact your sales chanel.

```
Start this routine?     [Y/N] <Y>
```

The ASM400 indicates the creation of the burning files and gives a completion message. Only the necessary files are created:

```
xxxxxx.BR1
xxxxxx.BR2
xxxxxx.BR3
xxxxxx.BR4
Burnfile(s) created !
```

8.2.2 Loading the User's Software into the EEPROMs

The ASM400 creates up to four burning files which are necessary for the burning of the EEPROM(s). Each of these files contains the 512 bytes of object code for one of the EEPROMs and the burning software in TSS400 object code. Before debugging can start, the EEPROMs have to be loaded with the users software.

The ADT400 emulator software is started by: C:/TSS400 <RETN>

The ADT400 asks if the last work is to be reloaded:

Do you wish to reload the last work ? NO <RETN>

Select "Load Version from File" out of the displayed menue. The ADT400 shows the existing versions, select "SSW.STA" which is the TSS400 Standard software version containing all necessary informations.

The ADT400 now asks for the wished task, select:

Emulation / Debugging <RETN> <ESC>
After this a selection of objects is shown, the "Wildcard" option is needed to get the "xxxxxx.BR1" object loaded: <F1> *.BR1 <RETN>
All objects with the extension "BR1" are shown now. The cursor movement keys may be used to select the "xxxxxx.BR1" object. After pressing of the return key, the "xxxxxx.BR1" object is loaded and reassembled. When this is finished the ADT400 aks:

Do you want to load Emulator processor data ? [Y/N] <N>

Pressing <SHIFT/F4> and then <SHIFT/F3> initializes the emulator. Pressing <F5> afterwards runs the burning program for the 1st EEPROM which lasts 2 seconds approximatively.

Pressing the <SPACE> bar halts the emulator and the display shows an endless BRANCH if the burning routine completed successfully.

If another EEPROM is to be burned, pressing of <ALT/L> shows the object selection seen before again. The "Wildcard" option is now to be used to get the "xxxxxx.BR2" files displayed: <F1> *.BR2 <RETN>

Proceeding is the same like shown for the burning of "xxxxxx.BR1" before. The same is to be done for the burning of "xxxxxx.BR3" and "xxxxxx.BR4" if existing.

8.2.3 Debugging the User's Software with the ADT400

Loading of the TSS400 Standard Software

After burning of the EEPROMs with the users software, the interpreter is to be loaded:

Pressing <ALT/L> shows an object selection. The file "SSW.OBJ" contains the interpreter. Pressing the <RETN> key after selecting this file with the cursor loads and reassembles the interpreter.

Pressing <SHIFT/F4> and then <SHIFT/F3> initializes the emulator.

Setting and Resetting Breakpoints

Pressing <F9> shows the table for selecting of the breakpoint data. At first all existing breakpoints have to be deleted:

```
TYPE:   Reset ALL   <RETN>
PAGE:   xxx
PC:     xxx
LABEL:  xxxxx
CMD:    xxxxx
```

The ADT400 asks: Clear all Breakpoints?   Y/N Y   <RETN>

The same procedure may be used for removing of one breakpoint.

For setting a breakpoint <F9> is to be pressed again. The breakpoint data appear on the screen, and the following inputs (here for the label DECODE) are made:

```
TYPE:   Label
PAGE:   xxx
PC:     xxx
LABEL:  DECODE   <RETN>
CMD:    xxxxx
```

Breakpoint Set

Pressing of <F5> starts the execution of the users software for one instruction byte or one instruction depending on the used breakpoint label. These two labels are explained with the debugging modes.

Debugging with the ADT400 allows the following modes:

- Single Byte Execution
- Single Step Execution
- Free Run
- Free Run with Software Breakpoints

The above mentioned debugging modes are described in the following:

Single Byte Execution

By setting an ADT400 breakpoint to the label BREAK inside the TSS400 Standard software, the Single Byte Execution is possible. This means, each time the breakpoint is reached, one single byte of the user's program was read and executed. Setting a breakpoint to the label BREAK and pressing <F5> will read and execute one instruction byte.

Single Step Execution

By setting an ADT400 breakpoint to the label MASTER inside the TSS400 Standard software, the Single Step Execution is possible. This means, each time the breakpoint is reached, one complete instruction with all parameter bytes was read and executed. Setting a breakpoint to DECODE and pressing <F5> will execute one instruction. The RAM shows the next instruction and address which is not yet executed.

Free Run

The user's program may be checked during Free Run by watching the display and the outputs. This mode is only possible after removing of all software and ADT400 breakpoints. Press <F5> after the initialization of the ADT400 to get Free Run.

Free Run with Software Breakpoints

The software breakpoints described later on are possible with the ADT400 too. See at "DEVELOPMENT WITH THE SOFTWARE SIMULATOR" how to use these software breakpoints. After resetting of all ADT400 breakpoints, press <F5> to get this run mode.

8.2.4 RAM Map of the TSS400 Standard

When a breakpoint is reached, the ADT400 shows the state of all registers of the TSS400. The RAM of the TSS400 is arrranged as follows.

Note: The shown RAM is configured like the RAM shown on the ADT400 screen. The numbers above the RAM are the Y-Register values, the numbers to the left of the RAM are the X-Register values shown by the ADT400.

Y	=0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X1		Ins old LSD	Instruction MSD	LSD	X MSB	PC	LSB	Subroutine Level 1			Subroutine Level 2			Subroutine Level 3		

Y	=0 7	8	9	10	11	12	13	14	15
X2	Storage Register 5 10E7 .... 10E0	Sign				3 2 1 0	7 Group 1 Flags	11	15 14 13 12

Y	=0 7	8	9	10	11	12	13	14	15
X3	Storage Register 4 10E7 .... 10E0	Sign		Counter 1 10E1	10E0	H3 H2 H1 0	H7 Group 1 Flags	ST LT	15 14 13 12

Y	=0 7	8	9	10	11	12	13	14	15
X4	Storage Register 3 10E7 .... 10E0	Sign				Short Timer MSD		Long Timer MSD	

Y	=0 7	8	9	10	11	12	13	14	15
X9	Storage Register 2 10E7 .... 10E0	Sign		Counter 2 10E1	10E0	Short Timer Buffer MSD		Long Timer Buffer MSD	

Y	=0 7	8	9	10	11	12	13	14	15
XA	Storage Register 1 10E7 .... 10E0	Sign				R1/K8 R1/K4 R1/K2 R1/K1	R2/K8 Keyboard	R3/K8	R4/K8 R4/K4 R4/K2 R4/K1

Y	=0 7	8	9	10	11	12	13	14	15
XB	Storage Register 0 10E7 .... 10E0	Sign							

Y	=0 7	8	9	10	11	12	13	14	15
XC	FLAC 10E7 .... 10E0	Sign	POS ZERO NEG		MSD		ADC Buffer (used as a HELP register too)		LSD

Y	=0 7	8	9	10	11	12	13	14	15
XF	REGB 10E7 .... 10E0	Sign	Identity A	5			MSD	ADC	

Figure 37: RAM Map of the TSS400 Standard

8.3 Development with the Software Simulator

The Software Simulator, which runs on all IBM-AT compatible Personal Computers (PC), allows fast development of the software for the TSS400 Standard. All functions, with the exception of the hardware communication with the inputs and outputs, are possible. For the development of the program's algorithms, no hardware at all is necessary.

All internal registers, inputs, outputs, flags, are shown simultaneously on one screen and may be modified whenever wished, also during the run mode.

The following part does not describe the handling of the Software simulator but gives some hints how to test the user's program with the Development Board. For a detailed description of the Software Simulator see "TSS400 Standard Simulator User's Guide".

Necessary components for the software development with the Software Simulator are:

- IBM-AT compatible Personal Computer
- ASM400 software installed on the Personal Computer
- software Simulator installed on the Personal Computer
- interface hardware for the connection of the Development Board
- development Board with EEPROM(s) and TSS400 inserted

The last two items are only needed if tests within the target hardware are necessary. For the test of the user's software, it is only necessary to assemble it with the ASM400 crossassembler. The produced object is loaded and executed bv the Software Simulator.

8.3.1 Restrictions of the Software Simulator

The Software Simulator is a software tool written in PASCAL, which simulates only the TSS400 Standard. This prohibits some functions of the TSS400 Standard. The following list gives a short overview concerning the differences:

- The R-Outputs, the I/O Pin and the K-Port are not available as outputs.
- The K-Port used as an input may be modified by Simulator commands only.
- The instructions do not work in exact realtime, only the timers do this.
- The instruction SLV is interpreted as a NOP.
- If overflow occurs during a multiplication, the FLAC is loaded with 99999999 due to the used PASCAL software. The TSS400 Standard looses the MSDs instead and holds only the 8 LSDs of the result.

8.3.2 Assembling the User's Software

The assembly of the users program is made exactly like described at "Development with the ADT400". See there how to proceed.

After completion of the assembly run, the ASM400 asks if the EEPROM burning files are wished. They are not needed, so the answer is NO.

```
Transferroutine:    → BURN.BR#
Start this routine?  [Y/N] <N>
```

The assembler ceases after the NO input.

8.3.3 Debugging the User's Software with the Software Simulator

The object file generated by the ASM400 crossassembler can be loaded into the Software Simulator's object buffer and tested until the complete algorithms are checked. See "TSS400 Standard Simulator User's Guide" for a description of the possibilities the Software Simulator gives for software testing.

8.3.4 Realtime Debugging of the User's Software

After verification of all software parts which do not need connection to the target hardware, the realtime tests with the Development Board connected to the target hardware may be started.

The Development Board is connected to the CENTRONICS interface connector with the delivered cable. The tested user's program is burned into the EEPROMs with the appropriate Simulator instruction and reread for verification. The user's program, stored in the EEPROMs now, may be started and stopped by instructions of the Software Simulator.

Realtime debugging with the Development Board is made by inserting software breakpoints into the user's program at locations where program parts are completed. These may be computations, measurement parts, keyboard tests and so on. The software breakpoints are inserted by the editor and are deleted after use or replaced by NOP instructions.

Several possibilities exist for halting and checking at those locations:

- jumps to the same location
- waiting for a definitive key to be pressed
- displaying of the register which contains important information
- displaying of the interesting registers with wait cycles

The possibilities mentioned above are described in the following:

Jumps to the same location:

At the locations where the program should cease, a jump to the same location is inserted:

```
             ...                    END OF PART TO BE CHECKED
L$555        JMP         L$555      ENDLESS LOOP
```

Example 43: Software breakpoint 1

The endless loop stops the program at location L$555. The outputs may be observed for correctness, the displayed information, too.

To proceed from this breakpoint, a change of the program with the editor, an assembly run and an EEPROM burning is necessary.

Waiting for a definitive key tc be pressed:

At the locations where the program should cease, a call to a subroutine is inserted. This subroutine waits until a certain (normally unused) key is pressed. After this activation, the program continues until the subroutine is invoked the next time.

After completion of the debugging, the subroutine and all calls to it are removed from the user's program.

Example: It shows the use and the code of the subroutine:

```
          ...                   PROGRAM PART 1 TO BE CHECKED
          CALL      SWBKPT      WAIT HERE UNTIL KEY IS PUSHED
          ...                   PROGRAM PART 2
          CALL      SWBKPT      WAIT HERE UNTIL KEY IS PUSHED
          ...                   PROGRAM PART 3
          CALL      SWBKPT      WAIT HERE UNTIL KEY IS PUSHED
          ...                   PROGRAM PART 4

* THIS ROUTINE IS ADDED DURING DEBUGGING. THE KEY TO BE
* ACTIVATED MAY BE CHOSEN FREE WITH THE TSTKEY-OPERAND
* THE USED KEY OF THE EXAMPLE IS CONNECTED TO R1 AND K1
*
SWBKPT    TSTKEY    >00+>C+0    R1, POS. STROBE, K1
          JP        SWRET       LO-HI TRANSITION ?
          JMP       SWBKPT      NO, WAIT
SWRET     RETN                  YES, PROCEED
```

Example 44: Software breakpoint 2

The used R-output/K-input combination may be defined with the operand of the TSTKEY instruction. The same is true for the strobe used with the R-output.

Displaying the Register which contains important information:

This kind of information getting may be used with the above mentioned software breakpoints. Any register with important information may be displayed before the software breakpoint occurs.

Example: The contents of the FLAC are important at the software breakpoint. Its contents (10E3 - 10E0) should be displayed then:

```
          ...                   PROGRAM PART TO BE CHECKED
          DISPLFL    >47        DISPLAY RESULT IN FLAC
          BYTE       >80        DIGIT 4  10E3
          BYTE       >80
          BYTE       >80
          BYTE       >80        DIGIT 7  10E0
L$556     JMP        L$556      STOP HERE
```

Example 45: Software breakpoint with display

With multiple key activations, the contents of some registers may be displayed in sequence after a software breakpoint. After each key activation the displaying of a Storage Register follows.

Example: The FLAC is to be displayed when the software breakpoint is reached. The activation of the key, connected to R3/K4 has to display the contents of STO1, STO4 and STO3 in sequence.

```
          ...                   PROGRAM PART TO BE CHECKED
          CALL       DISFLAC    DISPLAY FLAC
          CALL       SWBKPT     WAIT HERE UNTIL KEY IS PUSHED
          MOVSTOFL   1
          CALL       DISFLAC    DISPLAY STORAGE REGISTER 1
          CALL       SWBKPT     WAIT HERE UNTIL KEY IS PUSHED
          MOVSTOFL   4
          CALL       DISFLAC    DISPLAY STORAGE REGISTER 4
          CALL       SWBKPT     WAIT HERE UNTIL KEY IS PUSHED
          MOVSTOFL   3
          CALL       DISFLAC    DISPLAY STORAGE REGISTER 3
          ...                   PROCEED TO NEXT BREAKPOINT

* THE USED KEY IS CONNECTED TO R3 AND K4
*
SWBKPT    TSTKEY     >20+>C+2   R3, POS. STROBE, K4
          JP         SWRET      LO-HI TRANSITION ?
          JMP        SWBKPT     NO, WAIT
SWRET     RETN                  YES, PROCEED
*
* SUBROUTINE DISPLAYS FLAC AS IT IS
*
DISFLAC   DISPLFL    >17        DISPLAY DIGIT 1 - 7
          BYTE       >80        DIGIT 1
          BYTE       >80
          BYTE       >80
          BYTE       >80
          BYTE       >80
          BYTE       >80
          BYTE       >80        DIGIT 7
          ETN
```

Example 46: Software breakpoint with multiple display

Displaying the interesting registers with wait cycles:

Using one of the timers with the Done Mode gives the possibility to display one or more registers for a given time and to proceed then without any activity. If the wait time is long enough, the results may be written down conveniently for later checks.

Example: The FLAC is to be displayed first when the software breakpoint is reached. Then the contents of STO1, STO4 and STO3 have to be displayed in sequence. Each register display should last for six seconds.

```
          ...                 PROGRAM PART TO BE CHECKED
          CALL      DISFLAC   DISPLAY FLAC FOR 6 SECONDS
          CALL      WAIT6S
          MOVSTOFL  1
          CALL      DISFLAC   DISPLAY STORAGE REGISTER 1
          CALL      WAIT6S    WAIT FOR 6 SECONDS
          MOVSTOFL  4
          CALL      DISFLAC   DISPLAY STORAGE REGISTER 4
          CALL      WAIT6S    WAIT FOR 6 SECONDS
          MOVSTOFL  3
          CALL      DISFLAC   DISPLAY STORAGE REGISTER 3
          CALL      WAIT6S
          ...                 PROCEED TO NEXT BREAKPOINT

* WAITING ROUTINE 6 SECONDS
*
WAIT6S    LDTIML    6         LOAD LONG TIMER  WITH 6 SECONDS
          DONE                SLEEP FOR THIS TIME
          RETN                RETURN TO CALLER
*
DISFLAC   ...                 SUBROUTINE: SEE EXAMPLE ABOVE
```

Example 47: Software breakpoint with wait cycles

8.3.5 Most often occuring difficulties

User programs running errorfree with the Software Simulator sometimes show unexpected behaviour when running on the Development Board. This is due to the differences of the Software Simulator and the Development Board. A list with often occuring errors is given together with explanations and hints.

- no wake-up occurs	see chapter 1, 4
- wake-up occurs with wrong timing	see chapter 1, 3
- DONE mode is not executed	see chapter 4, 3
- ADC result is always >000	see chapter 2, 5, 8
- ADC result is always >FFF	see chapter 5, 8
- wrong results of computations	see chapter 6, 7, 8
- counters count wrong	see chapter 6
- no wake-up from Off or Done mode	see chapter 9
- wrong decimal points appear in LCD	see chapter 10

Reasons for the malfunctions:

1. **DL12 not set:** DL12 must be set, if the Short Timer is used. The timer handling of the interpreter bases on time differences, if DL12 is reset, the wake-up occurs every second with avalue of 0 contained in the 16 Hz hardware timer. The interpreter doesn't see a difference to the last value read and doesn't execute the instruction after the DONE instruction therefore. Setting of DL12 avoids this malfunction.

2. **DL13 not set:** DL13 must be set, if the current source of the ADC is used. If all measured values are >000, the not set DL13 may be the cause. Only if DL13 is set, a current is output to the addressed analog input An.

3. **Unused timers not stopped:** If the former executed program used one of the timers, wrong timing will occur with the currently running program. The reason is, neither the INIT nor the LOAD command do stop the timers (see INIT conditions). Inclusion of STPTIMS or STPTIML into the initialization part will cease this behaviour.

4. **Timer flags not reset:** The DONE cannot be entered as long as one of the timer flags (Group 1 Flags 10, 11) are set. The user's program must reset these flags after the verification that one of the timers woke-up the TSS400 Standard.

5. **No sensors or sensors at wrong analog input:** If an anolog input is measured which doesn't have a sensor connected, the result is as follows:

   >000: if DL13 is reset
   >FFF: if DL13 is set

   The wrong addressing may be caused by the operand of the MEASR instruction: The operand n addresses the analog input An+1.

6. **Erroneous use of the ">" sign:** The ASM400 assembler converts all operands into the hexadecimal format. This means where hexadecimal numbers are used (timers, display instructions) the operands may be defined in decimal or hexadecimal format. If BCD format is necessary (counters, loading of registers) a ">" sign must precede the number which instructs the assembler to take the number as it is. A missing ">" sign would yield the hexadecimal number for the operand (e.g. 2C for 44).

7. **Overflow occured:** No indication is given, if the result of an instruction overflows the FLAC. A worst case computation concerning the largest possible results must be made preceding the software development to avoid this error. SHIFTRn or ROUNDn may be used.

8. **Erroneous hardware design:** The values of the external hardware (Rext, Rsensor aso.) are not correct. Checking of the values with worst case design methods will avoid this error.

9. **DL8 set:** DL8 must be reset, if a wake-up is needed from the K-Port.

10. **Group 1 Flag bits set:** If the H-segment flag bits in Group 1 are set from a previous program, the according H-segments will be on in the LCD. Clearing of the flags will solve the problem.

9 Controlling of the TSS400 Standard in Slave Mode

It is possible to use the TSS400 Standard as a pure slave processor. The complete control is made by a host processor which sends instructions to the TSS400 Standard via the I/O Pin. The TSS400 executes these instructions and sends the results, if wished by the host, back also via the I/O Pin. The line between the TSS400's I/O Pin and the host is referred to as data line.

The host is connected to the TSS400 Standard with 3 lines:

- Data line: From an I/O Port to the TSS400's I/O Pin.
- Init line: From an output to the TSS400's INITN Pin.
- Common ground line.

If the TSS400 Standard is used in Slave Mode, it is not possible to use a program EEPROM. If an EEPROM is necessary, it must be connected to the host directly. R0 and R7 cannot be used in the Slave Mode.

The Slave mode can be used for:

- transferring of ADC values to the host for computations
- writing of computed variables to the RAM
- transferring of internal flags to the host and back
- initiation of measurements with defined parameters
- reading and writing of counters and timers

The reading and writing of the RAM enables the user to have complete informations concerning the state of the TSS400 system.

Due to the relative low speed of the TSS400 the transfers via the data line are determined as follows:

- controlling is always made by the host computer
- clocking is always made by the TSS400
- the TSS400 is always slave, the host is always master

All inputs or outputs start with the MSB of the MSD, followed by the MSB-1 of the MSD and end with the LSB of the LSD.

Note: All signals in the following parts are described as they appear at the TSS400 pins. The Development Board and possibly the users final hardware have inverting stages in between the TSS400 and the host's interface. So the signals may be inverted in reality.

9.1 Handshaking and Program Control

The I/O Pin is not only used for the transfer of instructions and data, but for the necessary handshaking, too. The principle is simple: Host and TSS400 Standard hold low the data line as long as they are busy. Only if both of them are ready for a transfer, the line is pulled up by the pullup resistor. When this occurs, the TSS400 Standard asks for the next instruction or sends data if this was wished by the host instruction received before (SLV eg.).

This allows:
- the host to evaluate the last received results and to prepare the next instruction for the TSS400 Standard.
- the TSS400 Standard to execute the received instruction.

For program control, the opcodes of the instructions shown in Figure 23 and 24 are transferred to the TSS400 Standard, with the MSB of the eight bit opcode first. If an instruction consists of more than one byte, the additional bytes have to be transferred when demanded by the TSS400 Standard.

9.2 Starting of the Slave Mode

The I/O Pin is used normally for the data transfers from and to the EEPROM. This makes a special interfacing necessary to start the Slave Mode.

- The host pulls down the INITN Pin to V_{SS}.
- The host releases the INITN Pin to V_{DD}, the TSS400 starts execution.
- The TSS400 resets the I/O Pin to LO for at least 75 us.
- The TSS400 switches the I/O Pin to input direction and waits until the host releases the line, too.

- When the host releases the I/O Pin to HI, the TSS400 Standard outputs 4 x 4 pulses, MSB first, which have to be prolonged (1s) or not (0s) by the host for the codes >FF and >A5 in sequence to initiate the Slave Mode. All other received codes initiate the normal EEPROM Mode.

Note: The first two nibbles, containing >FF should be used by the host for the measurement of the oscillator frequency of the TSS400. One bit is 24 internal instructions in length. The knowledge of the oscillator frequency is necessary for a good synchronization.

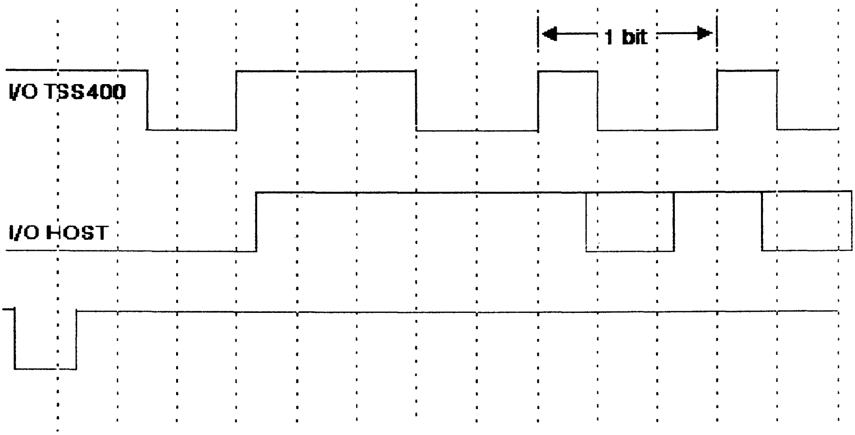


Figure 38: Mode check after initialization

The signals described before are shown below. The signals output by the host and the TSS400 are shown on separated lines although they appear both on the same line. The following figure shows the start of the TSS400 Standard software after an initialization:

9.3 Data Transfer

Data means instructions, operands, parameters, results and status information transferred between the host and the TSS400 Standard.

9.3.1 Data Transfer to the TSS400 Standard

The TSS400 Standard switches the data line to the input direction and waits until the host releases the line too, which leads to a high level. Then the TSS400 Standard reads in the next instruction or data to be processed. The host extends the pulses coming from the TSS400 if ones are to be transferred and doesn't extend the pulses if zeroes are to be transferred to the TSS400 Standard.

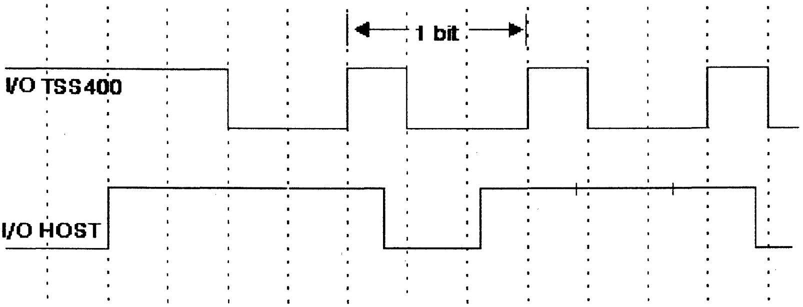


Figure 39: Data transfer to the TSS400 Standard

Figure 39 shows the first 3 bits of a data transfer after a busy host released the data line. The shown, transferred info is 010x.

Figure 40 shows the timing of a nibble transferred to the TSS400 Standard. The numbers show the length of the signals measured in TSS400 instructions. A TSS400 instruction needs 6/f_{OSC} seconds for the execution. The TSS400 scans the data line 12 instructions after the positive edge of the pulse.

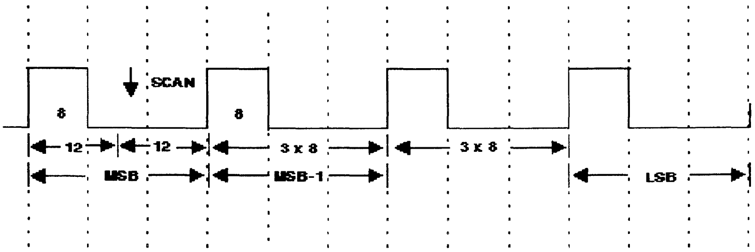


Figure 40: I/O Timing of a data transfer to the TSS400 Standard

Writing of the complete RAM

The instruction SLV allows the writing of the complete RAM. The transfer starts with the MSB of the nibble Y = 0 located in the X1-bank and ends with the LSB of the nibble Y = 15 located in the XF-bank. See the RAM map shown in Figure 37 for the meaning of the bits. The RAM map also shows the sequence, the X-Banks are read in.

576 bits are read by the TSS400 Standard ($9 \cdot 16 \cdot 4 = 576$)

Warning: It is strongly advisable to read out and store the RAM before any RAM writing is made. Non predictable results will occur, if RAM parts are not restored exactly like they were read out. Only the nibbles to be changed should be modified.

9.3.2 Data Transfer to the Host

If an instruction assumes results from the TSS400 Standard, the TSS400 sends the data after the host released the data line.

The timing for one nibble (which contains 5) is shown in Figure 41. The numbers below the figure show the length measured in TSS400 instructions. A TSS400 instruction needs $6/f_{OSC}$ seconds.

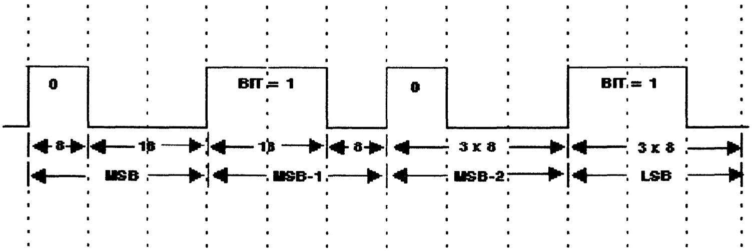


Figure 41: I/O Timing of a data transfer to the Host

Reading of the complete RAM

The instruction SLV allows the reading of the complete RAM. The transfer starts with the MSB of the nibble Y = 0 located in the X1-bank and ends with the LSB of the nibble Y = 15 located in the XF-bank. See the RAM map shown in Figure 37 for the meaning of the bits. The RAM map shows also the sequence, the X-Banks are read out.

576 bits are written by the TSS400 Standard ($9 \cdot 16 \cdot 4 = 576$)

9.4 Idle Modes for the Slave

The host can deactivate the TSS400 Standard if activity is not needed. Two possibilities exist:

- Holding low the I/O Pin line after the last instruction.
- Loading of one of the timers with a constant and usage of the Done Mode afterwards.

The first method uses the handshake feature of the TSS400 Standard. Normal operating current is drawn from the power supply.

The second method has the advantage of the very low supply current drawn during the Done Mode.

Example: The TSS400 is not needed for 6 seconds. The host sends the two instructions shown below to the TSS400 Standard which sleeps for 6 seconds. When this time elapsed, the TSS400 Standard will call for the next instruction.

```
LDTIML    6          6 SECONDS DEACTIVATION
DONE                 DONE MODE
...                  CALL FOR NEXT INSTRUCTION
```

Example 48: Deactivation of the Slave by the Done Mode

9.5 Not Executable Instructions

The following instructions cannot be executed in the Slave Mode:

MOVFLPRM no EEPROM connected to the TSS400 in Slave Mode
MOVPRMRB same as above

The program flow instructions like CALL, JMP and so on are senseless in the Slave Mode.

APPENDIX A

Hardware Definition

For references the actual hardware definition of the TSS400 Standard, used for production purposes, is listed below:

```
!
.OPTION VERSION
VERSION S220
!
.OPTION PINNING
!                    NO TEST     TEST   K       R      S/C
!                    KC = 0            RMIN    RMOUT   HVT
PAD  1  IS-PIN  1   APPL VSS      TEST VSS     VSS     VSS
!
.OPTION ROM
TSS0400  OBJECT:STDSW   CREATED 11 NOV 1988 11:11:11 VERS J.8 LB
0046000D002100400126014700E2000D0000008C008D008300000127003601 9C
0098000001000 11E004501800000 01E2019C000000000004500980098008 80044
```

Note: Only four lines out of the 128 object lines are shown.

```
01E3009D008D017601150067000001FA001A008400000036018901490048 0100
00BF005A006B009C01BC00060056008001 4C01FA004F0115009400230 1E800BF
!
.OPTION K1K2K4K8IO
INPUT K1  PULL-DOWN  NONE   ; NO PULL-DOWN ANYWHERE
INPUT K2  PULL-DOWN  NONE
INPUT K4  PULL-DOWN  NONE
INPUT K8  PULL-DOWN  NONE
INPUT I/O PULL-DOWN  NONE
OUTPUT K1 OPEN-SOURCE NO    ; PUSH-PULL OUTPUTS
OUTPUT K2 OPEN-SOURCE NO    ; PUSH-PULL OUTPUTS
OUTPUT K4 OPEN-SOURCE NO    ; PUSH-PULL OUTPUTS
OUTPUT K8 OPEN-SOURCE NO    ; PUSH-PULL OUTPUTS
!
.OPTION K8FF.
FLIP-FLOP DISABLED          ; NO FLIP FLOP IN K8-INPUT
!
.OPTION ADC-RANGE
RANGE-IS LARGE              ; 0.1013 x SVDD - 0.4946 x SVDD
!
.OPTION WAKE-UP-FREQUENCY
FREQUENCY 16HZ              ; 16 Hz WAKEUP FREQUENCY FOR DL12 = 1
!
.OPTION CLOCK
```

```
FREQUENCY NONE              ; CLOCK NOT PINNED OUT
!
.OPTION TIMOSC
USE TXOSC                   ; QUARTZ IS USED FOR TIMER
!
.OPTION COMMON
USE MUX4                    ; 4 COMMON MULTIPLEX
!
.OPTION SELECT-LINES        ; ADDRESSING OF THE SELECT LINES
LOAD  S1+S2    BY  Y1
LOAD  S3+S4    BY  Y2
LOAD  S5+S6    BY  Y3
LOAD  S7+S8    BY  Y4
LOAD  S9+S10   BY  Y5
LOAD  S11+S12  BY  Y6
LOAD  S13+S14  BY  Y7
!                    COMMON  1 2 3 4
FOR  MUX4  AND  S2N-1  LOAD  C F H E  ; FAIL SAFE
FOR  MUX4  AND  S2N    LOAD  A B D G  ; CONFIGURATION
!
.OPTION SEGMENT-PLA
!        DL14 = 0 , ST = 0
CHARACTER  0 HAS-SEGMENTS A B C D E F ;         0
CHARACTER  1 HAS-SEGMENTS B C ;                 1
CHARACTER  2 HAS-SEGMENTS A B D E G ;           2
CHARACTER  3 HAS-SEGMENTS A B C D G ;           3
CHARACTER  4 HAS-SEGMENTS B C F G ;             4
CHARACTER  5 HAS-SEGMENTS A C D F G ;           5
CHARACTER  6 HAS-SEGMENTS A C D E F G ;         6
CHARACTER  7 HAS-SEGMENTS A B C ;               7
CHARACTER  8 HAS-SEGMENTS A B C D E F G ;       8
CHARACTER  9 HAS-SEGMENTS A B C D F G ;         9
CHARACTER 10 HAS-SEGMENTS A B C E F G ;         A
CHARACTER 11 HAS-SEGMENTS C D E F G ;           b
CHARACTER 12 HAS-SEGMENTS A D E F ;             C
CHARACTER 13 HAS-SEGMENTS B C D E G ;           d
CHARACTER 14 HAS-SEGMENTS A D E F G ;           E
CHARACTER 15 HAS-SEGMENTS A E F G ;             F
!
!        DL14 = 0 , ST = 1
CHARACTER 16 HAS-SEGMENTS NONE   ;              BLANK
CHARACTER 17 HAS-SEGMENTS B C D ;               J
CHARACTER 18 HAS-SEGMENTS D E F ;               L
CHARACTER 19 HAS-SEGMENTS A B E F G ;           P
CHARACTER 20 HAS-SEGMENTS B C D E F ;           U
CHARACTER 21 HAS-SEGMENTS D E G ;               c
CHARACTER 22 HAS-SEGMENTS C E F G ;             h
CHARACTER 23 HAS-SEGMENTS D E ;                 l
CHARACTER 24 HAS-SEGMENTS C E G ;               n
CHARACTER 25 HAS-SEGMENTS C D E G ;             o
CHARACTER 26 HAS-SEGMENTS E G ;                 r
CHARACTER 27 HAS-SEGMENTS D E F G ;             t
CHARACTER 28 HAS-SEGMENTS C D E ;               v,u
CHARACTER 29 HAS-SEGMENTS B C D F G ;           Y
CHARACTER 30 HAS-SEGMENTS B C E F G ;           H
CHARACTER 31 HAS-SEGMENTS B C G ;               -1
!
!        DL14 = 1 , ST = 0                      COMBINATIONS
CHARACTER 32 HAS-SEGMENTS NONE ;                BLANK
CHARACTER 33 HAS-SEGMENTS A
CHARACTER 34 HAS-SEGMENTS F
CHARACTER 35 HAS-SEGMENTS A F
CHARACTER 36 HAS-SEGMENTS B
CHARACTER 37 HAS-SEGMENTS A B
CHARACTER 38 HAS-SEGMENTS B F
CHARACTER 39 HAS-SEGMENTS A B F
CHARACTER 40 HAS-SEGMENTS G ;                   MINUS SIGN
CHARACTER 41 HAS-SEGMENTS A G
CHARACTER 42 HAS-SEGMENTS F G
CHARACTER 43 HAS-SEGMENTS A F G
CHARACTER 44 HAS-SEGMENTS B G
CHARACTER 45 HAS-SEGMENTS A B G
CHARACTER 46 HAS-SEGMENTS B F G
CHARACTER 47 HAS-SEGMENTS A B F G ;             DEGREE
!
!        DL14 = 1 , ST = 1
CHARACTER 48 HAS-SEGMENTS E
CHARACTER 49 HAS-SEGMENTS A E
CHARACTER 50 HAS-SEGMENTS E F
CHARACTER 51 HAS-SEGMENTS A E F
CHARACTER 52 HAS-SEGMENTS B E
CHARACTER 53 HAS-SEGMENTS A B E
CHARACTER 54 HAS-SEGMENTS B E F
CHARACTER 55 HAS-SEGMENTS A B E F
CHARACTER 56 HAS-SEGMENTS E G
CHARACTER 57 HAS-SEGMENTS A E G
CHARACTER 58 HAS-SEGMENTS E F G
CHARACTER 59 HAS-SEGMENTS A E F G
CHARACTER 60 HAS-SEGMENTS C ;                   SEGMENT C
CHARACTER 61 HAS-SEGMENTS D ;                   SEGMENT D
CHARACTER 62 HAS-SEGMENTS A B C D ;             ]
CHARACTER 63 HAS-SEGMENTS A B C D E ;           J
.END                                  ; END OF HARDWARE OPTIONS
```

APPENDIX B

Applicable Documents

- TSS400 Application Report EB175 E
- TSS400 User's Guide
- TSS400 Software User's Guide
- TSS400 ADT User's Guide
- TSS400 Standard Simulator User's Guide
- 4-Bit Microcomputer Cross Assembler User's Guide

APPENDIX C

TSS400 Standard Instructions

Register to Register Transfer Instructions

MOVFLSTO	n	Move FLAC to Storage Register n
MOVSTOFL	n	Move Storage Register n to FLAC
MOVRBSTO	n	Move REGB to Storage Register n
MOVSTORB	n	Move Storage Register n to REGB
EXCHRBFL		Exchange REGB and FLAC Registers
MOVFLRB		Move FLAC to REGB
MOVRBFL		Move REGB to FLAC
MOVFLPRM	Label	Move FLAC to EEPROM starting at Address Label
MOVPRMRB	Label	Move EEPROM Contents starting at Address Label to REGB

Arithmetic Instructions (Results always to FLAC)

ADD	Add REGB to FLAC decimally
SUB	Subtract REGB from FLAC decimally
MPY	Multiply FLAC and REGB decimally

DIV	Divide FLAC by REGB decimally	
ADDH	Add REGB to FLAC hexadecimally	
SUBH	Subtract REGB from FLAC hexadecimally	
HEXDEC	Hexadecimal to decimal Conversion	
ROUNDn	Round FLAC n times	(0 < n < 5)
SHIFTRn	Shift right FLAC n times	(0 < n < 3)
SHIFTLn	Shift left FLAC n times	(0 < n < 3)

Arithmetics Compare Instructions

CMPFLRB	Compare FLAC and REGB. Set Status Flags.
TSTRB	Test Contents of REGB. Set Status Flags.

Bit Manipulation Instructions

SBIT	n	Set Flag Bit n	(0 ≤ n < 16)
RBIT	n	Reset Flag Bit n	(0 ≤ n < 16)
TBIT	n	Test Flag Bit n	(0 ≤ n < 16)
SELGRPn		Select Flag Bits Group n	(0 < n < 3)
OR		Or FLAC and REGB, Result to FLAC	
AND		And FLAC and REGB, Result to FLAC	

Constant Transfer Instructions

LDFLPOS	n	Load FLAC with a positive Constant contained in the n following Bytes in BCD Format (n = 0 : 4 Bytes).
LDFLNEG	n	Load FLAC with a negative Constant (Same as above).
LDRBPOS	n	Load REGB with a positive Constant.(Same as above).
LDRBNEG	n	Load REGB with a negative Constant.(Same as above).
CLRFL		Clear FLAC Register
CLRRB		Clear REGB

Input/Output Instructions

SETR	n	Set Output Rn (DLn)	(n # 0,7,14,15)
RSTR	n	Reset Output Rn (DLn)	(n # 0,7,14,15)
TSTKEY	>NN	Test Keyboard like described by operand	
KINTIM		Actualize K-Input and Timers	
KIN		Read K-Input to FLAC (LSD) and FLAGS 12-15	
FLKOUT		Output LSD of FLAC to K-Port	
KOUT	n	Output Constant n to K-PORT	

Program Flow Control Instructions

JMP	Label	Jump unconditional	
JZ	Label	Jump if zero	(ZERO = 1)
JEQ	Label	Jump if equal	(ZERO = 1)
JNZ	Label	Jump if not zero	(ZERO = 0)
JNE	Label	Jump if not equal	(ZERO = 0)
JP	Label	Jump if positive	(POS = 1)
JN	Label	Jump if negative	(NEG = 1)
CALL	Label	Call Subroutine	
RETN	Return from Subroutine		

Analog-Digital-Converter Instructions

SVDDON		Set Converter Supply Voltage
SVDDOFF		Reset Converter Supply Voltage
MEASR	a	Measure addressed A/D Input a+1. Following Byte contains number of conversions and Mode.
ADJBATT		Measure Battery Voltage, Result to FLAC
CHKBATT		Check Battery Voltage
ADJCOMP		Measure A/D Input, Result to FLAC
CHKCOMP		Compare A/D Input with Value in FLAC

Display Instructions

DISPLDGn	NN	Display Info of Operand >NN in Digit n.
DISPLCLR		Clear Display
DISPLFL	>MN	Display FLAC from Digit M to N. Appended (M-N+1) Bytes contain Info for each Digit.

Miscellaneous Instructions

DONE		Enter Done Mode
OFF		Enter Off Mode
NOP		No Operation
SLV	>NN	Host Control Instruction

Counter Instructions

DECCNTn		Decrement Counter n decimally
INCCNTn		Increment Counter n decimally
DECDBL		Decrement Double Counter decimally
INCDBL		Increment Double Counter decimally
LDCNTn	>NN	Load Counter n decimally with Constant >NN
MOVCNTn		Move Counter n to REGB
MOVDBL		Move Combined Counters to REGB

Timer Instructions

LDTIML	NNN	Load Long Timer with Constant NNN
LDTIMS	NNN	Load Short Timer with Constant NNN
ACTTIM		Actualize Timers
STPTIML		Stop Long Timer
STPTIMS		Stop Short Timer

TSS400 Family User's Guide

Sensor Signal Processors

TI Worldwide Sales Offices

ALABAMA: Huntsville: 4960 Corporate Drive, Suite N-150, Huntsville, AL 35805-6202, (205) 837-7530.

ARIZONA: Phoenix: 8825 N. 23rd Avenue, Suite 100, Phoenix, AZ 85021, (602) 995-1007; **Tucson:** 818 W. Miracle Mile, Suite 43, Tucson, AZ 85705, (602) 292-2640.

CALIFORNIA: Irvine: 17891 Cartwright Drive, Irvine, CA 92714, (714) 660-1200; **Roseville:** 1 Sierra Gate Plaza, Suite 255B, Roseville, CA 95678, (916) 786-9208; **San Diego:** 5625 Ruffin Road, Suite 100, San Diego, CA 92123, (619) 278-9601; **Santa Clara:** 5353 Betsy Ross Drive, Santa Clara, CA 95054, (408) 980-9000; **Woodland Hills:** 21550 Oxnard Street, Suite 700, Woodland Hills, CA 91367, (818) 704-8100.

COLORADO: Aurora: 1400 S. Potomac Street, Suite 101, Aurora, CO 80012, (303) 368-8000.

CONNECTICUT: Wallingford: 9 Barnes Industrial Park Road, Wallingford, CT 06492, (203) 269-0074.

FLORIDA: Altamonte Springs: 370 S. North Lake Boulevard, Suite 1008, Altamonte Springs, FL 32701, (407) 260-2116; **Fort Lauderdale:** 2950 N.W. 62nd Street, Suite 100, Fort Lauderdale, FL 33309, (305) 973-8502; **Tampa:** 4803 George Road, Suite 390, Tampa, FL 33634, (813) 885-7411.

GEORGIA: Norcross: 5515 Spalding Drive, Norcross, GA 30092, (404) 662-7900.

ILLINOIS: Arlington Heights: 515 W. Algonquin, Arlington Heights, IL 60005, (708) 640-3000.

INDIANA: Carmel: 550 Congressional Drive, Suite 100, Carmel, IN 46032, (317) 573-6400; **Fort Wayne:** 118 E. Ludwig Road, Suite 102, Fort Wayne, IN 46825, (219) 482-3311.

IOWA: Cedar Rapids: 373 Collins Road N.E., Suite 201, Cedar Rapids, IA 52402, (319) 395-9550.

KANSAS: Overland Park: 7300 College Boulevard, Lighton Plaza, Suite 150, Overland Park, KS 66210, (913) 451-4511.

MARYLAND: Columbia: 8815 Centre Park Drive, Suite 100, Columbia, MD 21045, (301) 964-2003.

MASSACHUSETTS: Waltham: 950 Winter Street, Suite 2800, Waltham, MA 02154, (617) 895-9100.

MICHIGAN: Farmington Hills: 33737 W. 12 Mile Road, Farmington Hills, MI 48018, (313) 553-1500; **Grand Rapids:** 3075 Orchard Vista Drive S.E., Grand Rapids, MI 49506, (616) 957-4202.

MINNESOTA: Eden Prairie: 11000 W. 78th Street, Suite 100, Eden Prairie, MN 55344, (612) 828-9300.

MISSOURI: St. Louis: 11816 Borman Drive, St. Louis, MO 63146, (314) 994-2100.

NEW JERSEY: Iselin: Parkway Towers, 485 E. Route 1 South, Iselin, NJ 08830, (201) 750-1050.

NEW MEXICO: Albuquerque: 1224 Parsons Court, N.E., Albuquerque, NM 87112, (505) 291-0495.

NEW YORK: East Syracuse: 6365 Collamer Drive, East Syracuse, NY 13057, (315) 463-9291; **Fishkill:** 300 Westage Business Center, Suite 140, Fishkill, NY 12524, (914) 897-2900; **Melville:** 1895 Walt Whitman Road, P.O. Box 2936, Melville, NY 11747, (516) 454-6600; **Pittsford:** 2851 Clover Street, Pittsford, NY 14534, (716) 385-6770.

NORTH CAROLINA: Charlotte: 8 Woodlawn Green, Suite 100, Charlotte, NC 28217, (704) 527-0930; **Raleigh:** 2809 Highwoods Boulevard, Suite 100, Raleigh, NC 27625, (919) 876-2725.

OHIO: Beachwood: 23775 Commerce Park Road, Beachwood, OH 44122, (216) 464-6100; **Beavercreek:** 4200 Colonel Glenn Highway, Suite 600, Beavercreek, OH 45431, (513) 427-6200.

OREGON: Beaverton: 6700 S.W. 105th Street, Suite 110, Beaverton, OR 97005, (503) 643-6758.

PENNSYLVANIA: Blue Bell: 670 Sentry Parkway, Blue Bell, PA 19422, (215) 825-9500.

PUERTO RICO: Hato Rey: 615 Merchantile Plaza Building, Suite 505, Hato Rey, PR 00918, (809) 753-8700.

TENNESSEE: Johnson City: P.O. Drawer 1255, Erwin Hwy., Johnson City, TN 37605, (615) 461-2192.

TEXAS: Austin: 12501 Research Boulevard, Austin, TX 78759, (512) 250-7655; **Dallas:** 7839 Churchill Way, Dallas, TX 75251, (214) 917-1264; **Houston:** 9301 Southwest Freeway, Commerce Park, Suite 360, Houston, TX 77074, (713) 778-6592.

UTAH: Salt Lake City: 1800 S. West Temple Street, Suite 201, Salt Lake City, UT 84115, (801) 466-8973.

WASHINGTON: Redmond: 5010 148th Avenue N.E., Building B, Suite 107, Redmond, WA 98052, (206) 881-3080.

WISCONSIN: Waukesha: 20825 Swenson Drive, Suite 900, Waukesha WI 53186, (414) 798-1001.

CANADA: Nepean: 301 Moodie Drive, Mallorn Center, Suite 102, Nepean, Ontario, Canada K2H 9C4, (613) 726-1970; **Richmond Hill:** 280 Centre Street East, Richmond Hill, Ontario, Canada L4C 1B1, (416) 884-9181; **St. Laurent:** 9460 Trans Canada Highway, St. Laurent, Quebec, Canada H4S 1R7, (514) 335-8392.

ARGENTINA: Texas Instruments Argentina Viamonte 1119, 1053 Capital Federal, Buenos Aires, Argentina, 1/748-3699.

AUSTRALIA (& NEW ZEALAND): Texas Instruments Australia Ltd., 6-10 Talavera Road, North Ryde (Sydney), New South Wales, Australia 2113, 2 887-1122; 5th Floor, 418 Street, Kilda Road, Melbourne, Victoria, Australia 3004. 3 267-4677; 171 Philip Highway, Elizabeth, South Australia 5112, 8 255-2066.

AUSTRIA: Texas Instruments GmbH., Hietzinger Kai 101-105, A-1130 Wien, (0222) 9100-0.

BELGIUM: S.A. Texas Instruments Belgium N.V., 11, Avenue Jules Bordetlaan 11, 1140 Brussels, Belgium, (02) 242 30 80.

BRAZIL: Texas Instruments Electronicos do Brasil Ltda., Rua Paes Leme, 524-7 Andar Pinheiros, 05424 Sao Paulo, Brazil, 0815-6166.

DENMARK: Texas Instruments A/S, Marielundvej 46E, 2730 Herlev, Denmark, (42) 91 74 00.

FINLAND: Texas Instruments OY, Ahertajantie 3, P.O. Box 81, 02101 Espoo, Finland, (90) 461-422.

FRANCE: Texas Instruments France, 8-10 Avenue Morane Saulnier-B.P. 67, 78141 Velizy Villacoublay Cedex, France, (1) 30 70 10 03.

GERMANY: Texas Instruments Deutschland GmbH., Haggertystrasse 1, 8050 Freising-RFA, (08161) 80-0 od. Nbst; Kurfurstendamm 195-196, 1000 Berlin 15, (030) 8 82 73 65; Dusseldorfer Strasse 40, 6236 Eschborn 1, (06196) 80 70; III, Hagen 43/Kibbelstrasse 19, 4300 Essen 1, (0201) 24 25-0; Kirchhorster Strasse 2, 3000 Hannover 51, (0511) 64 68-0; Maybachstrasse II, 7302 Ostfildern 2 (Nellingen), (0711) 34 03-0.

HOLLAND: Texas Instruments Holland B.V., Hogehilweg 19, Postbus 12995, 1100 AZ Amsterdam-Zuidoost, Holland, (020) 5602911.

TEXAS
INSTRUMENTS

HONG KONG: Texas Instruments Hong Kong Ltd., 8th Floor, World Shipping Center, 7 Canton Road, Kowloon, Hong Kong, 7351223.

IRELAND: Texas Instruments Ireland Ltd., 7/8 Harcourt Street, Dublin 2, Ireland, (01) 75 52 33.

ITALY: Texas Instruments Italia S.p.A., Centro Direzionale Colleoni, Palazzo Perseo-Via Paracelso 12, 20041, Agrate Brianza (Mi), (039) 63221; Via Castello della Magliana, 38, 00148 Rome, (06) 5222651; Via Amendola, 17, 40100 Bologna, (051) 554004.

JAPAN: Texas Instruments Japan Ltd., Aoyama Fuji Building 3-6-12 Kita-aoyama Minato-ku, Tokyo, Japan 107, 03-498-2111; MS Shibaura Building 9F, 4-13-23 Shibaura, Minato-ku, Tokyo, Japan 108, 03-769-8700; Nissho-iwai Building 5F, 2-5-8 Imabashi, Chuou-ku, Osaka, Japan 541, 06-204-1881; Dai-ni Toyota Building Nishi-kan 7F, 4-10-27 Meieki, Nakamura-ku, Nagoya, Japan 450, 052-583-8691; Kanazawa Oyama-cho Daiichi Seimei Building 6F, 3-10 Oyama-cho, Kanazawa, Ishikawa, Japan 920, 0762-23-5471; Matsumoto Showa Building 6F, 1-2-11 Fukashi, Matsumoto, Nagano, Japan 390, 0263-33-1060; Daiichi Olympic Tachikawa Building 6F, 1-25-12, Akebono-cho, Tachikawa, Tokyo, Japan 190, 0425-27-6760; Yokohama Nishiguchi KN Building 6F. 2-8-4 Kita-Saiwai, Nishi Ku, Yokohama, Kanagawa, Japan 220, 045-322-6741; Nihon Seimei Kyoto Yasaka Building 5F, 843-2, Higashi Shiokohjicho, Higashi-iru, Nishinotoh-in, Shiokohji-dori, Shimogyo-ku, Kyoto, Japan 600, 075-341-7713; Sumitomo Seimei Kumagaya Building 8F, 2-44 Yayoi, Kumagaya, Saitama, Japan 360, 0485-22-2440; 2597-1, Aza Harudai, Oaza Yasaka, Kitsuki, Oita, Japan 873, 09786-3-3211.

KOREA: Texas Instruments Korea Ltd., 28th Floor, Trade Tower, 159, Samsung-Dong, Kangnam-ku Seoul, Korea, 2 551 2800.

MEXICO: Texas Instruments de Mexico S.A., Alfonso Reyes 115, Col. Hipodromo Condesa, Mexico, D.F., Mexico 06120, 5/525-3860.

MIDDLE EAST: Texas Instruments, No. 13, 1st Floor Mannai Building, Diplomatic Area, P.O. Box 26335, Manama Bahrain, Arabian Gulf, 973 274681.

NORWAY: Texas Instruments Norge A/S, PB 106, Refstad (Sinsenveien 53), 0513 Oslo 5, Norway, (02) 155090.

PEOPLE'S REPUBLIC OF CHINA: Texas Instruments China Inc., Beijing Representative Office, 7-05 CITIC Building, 19 Jianguomenwai Dajie, Beijing, China, 500-2255, Ext. 3750.

PHILIPPINES: Texas Instruments Asia Ltd., Philippines Branch, 14th Floor, Ba-Lepanto Building, Paseo de Roxas, Makati, Metro Manila, Philippines, 2 817 6031.

PORTUGAL: Texas Instruments Equipamento Electronico (Portugal) LDA., 2650 Moreira Da Maia, 4470 Maia, Portugal (2) 948 1003.

SINGAPORE (& INDIA, INDONESIA, MALAYSIA, THAILAND): Texas Instruments Singapore (PTE) Ltd., Asia Pacific Division, 101 Thomson Road, #23-01, United Square, Singapore 1130, 350 8100.

SPAIN: Texas Instruments Espana S.A., c/Gobelas 43, Ctra de la Coruna km 14, La Florida, 28023, Madrid, Spain, (1) 372 8051; c/Diputacion, 279-3-5, 08007 Barcelona, Spain, (3) 317 91 80.

SWEDEN: Texas Instruments International Trade Corporation (Sverigefilialen), (visit address: Isafjordsgatan 7, Kista), Box 30, S-164 93 Kista, Sweden, (08) 752 58 00.

SWITZERLAND: Texas Instruments Switzerland AG, Riedstrasse 6, CH-8953 Dietikon, Switzerland, (01) 740 22 20.

TAIWAN: Texas Instruments Supply Company, Taiwan Branch, Room 903, 9th Floor, Bank Tower, 205 Tun Hwa N. Road, Taipei, Taiwan, Republic of China, 2 713 9311.

UNITED KINGDOM: Texas Instruments Ltd., Manton Lane, Bedford, England, MK41 7PA, (0234) 270 111.

Printed in Germany

TEXAS INSTRUMENTS

PRINTED IN GERMANY