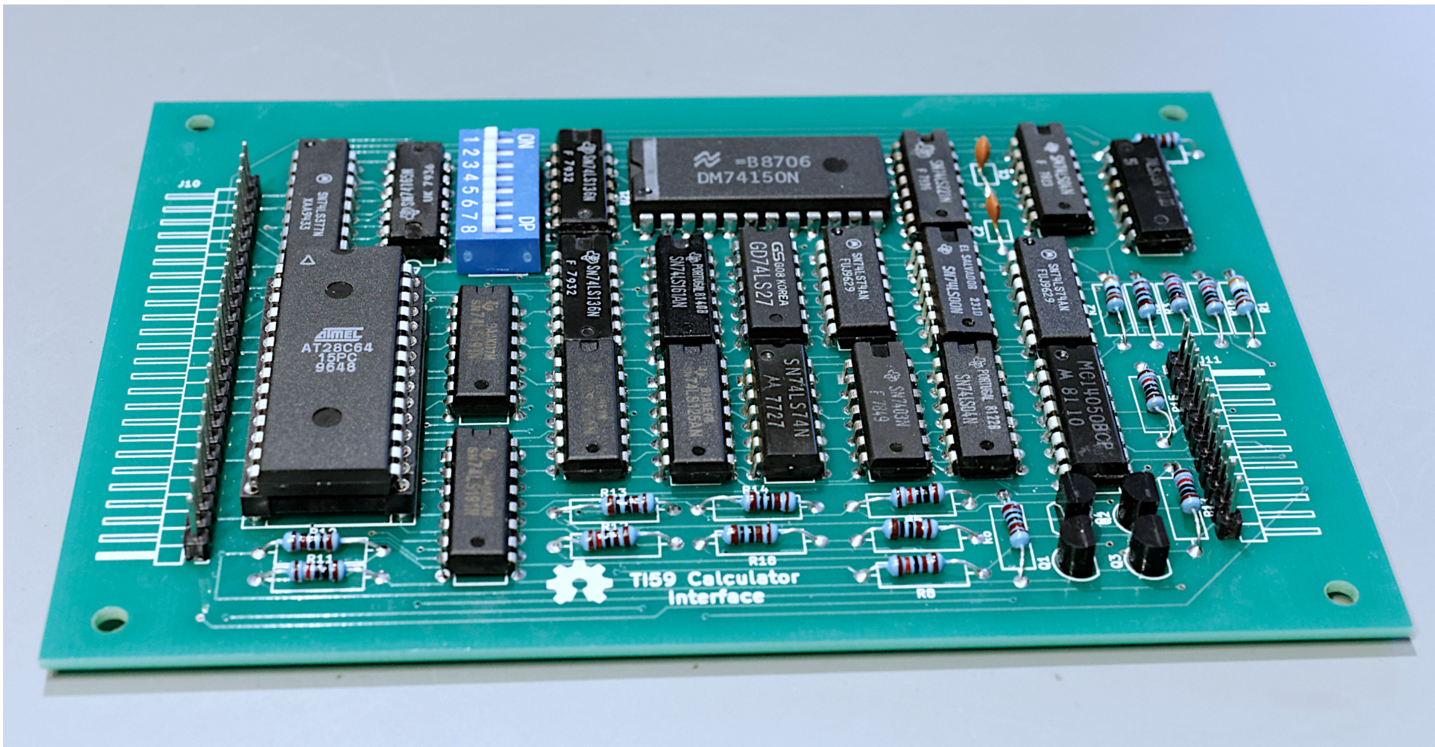
TI-59 INTERFACE

A universal interface for keyboard control
of a
Texas Instruments TI-59/TI-58
pocket calculator
using printer connection points
for digital control
without calculator modifications



by
Torben Rune

February 9. 1981

Table of Contents

Preface

In 1979 Lennart Schultz, student of engineering at the Technical University of Denmark, developed a high level language for TI-59 (TI-58) pocket calculators. The language was called TI-hill (Texas Instruments high level language), and it is a PASCAL-like language, that enables the user to write very powerful and well documented programs for TI-59 calculators.
In the first versions of the TI-hill compiler, the generated calculator code was simply printed out, and had to be keyed in by hand. This was indeed an extremely demanding job, not only because the amount of generated code increases drastic with the complexity of the program, but also because it is very difficult - faultlessly - to key in a program when you are unable to see how the program is intended to work.

On that background it was decided to try to build up an interface system that could enter the generated code into the calculator automatically.

In June 1980 the first attempts to construct such a system was made, but it soon became clear that it would be quite a hard job. The first attempt wrecked in lack of knowledge. After benevolent assistance from Texas Instruments it was possible to continue the construction of the interface, and during the summer vacation 1980 the interface system scripted in this paper was developed.

Since that, a 8080 MPU system has been equipped with a program system for handling keycode files generated by the TI-hill compiler, and the whole system has been used during the passed months at the university without errors. However minor changes must be made in the TI-hill compiler before the whole system is perfect.

I want to express my thanks to lecturer Henning Isaksson, who has been the coordinator of the project , and to Texas Instruments for benevolent collaboration.
Torben Rune Petersen

Hillerød, Denmark

February 9th, 1981

Characteristics of the interface

The primary purpose of the interface system is to enable the control a TI-59 (TI-58) pocket calculator. The system is constructed so that:

1. it is possible to connect the system to a computer, either a large computer system (if convenient by using a microprocessor system as slave unit) or a microcomputer.
2. The system should be designed with special reference to the developed TI-hill compiler, and
3. an arbitrary TI-59 (TI -58) calculator can be connected to the system. This demand is imposed because the magnetic card readers in TI-59 calculators are not necessarily compatible.

The developed interface system has the following properties:

The system can be controlled directly by a microprocessor. Special I/O logic has been adapted for a 8080A based processor system, but all processors can control the interface by means of relatively simple logic. The controlling of the calculator is performed by forcing signals on to the signal lines accessible from the printer connection in the bottom of the calculator. This means, that any TI- 59 (TI-58) calculator can be used with this system, as there are made no changes to the calculator itself.

A PC-100C printer (a dedicated Texas Instruments product) can be modified, so that the connection to the calculator can be made through the connection points in the printer. (See figure 5 for details).

The interface system can simulate complete keyboard operations. That means, that it is possible to control the calculator from the system exactly in the same way as if the operations was made form the keyboard of the calculator. This imply that it is possible to enter programs into the calculator, that all its calculating functions (arithmetic function etc.) can be used, so that the calculator can be used as a fully operational remote unit.

Use of the interface system

The interface is constructed so that its use becomes as simple as possible. This is achieved by laying the control functions in the hardware system rather than in system software, so that the user only has to send keycodes to the system, without worrying about the function of the interface. The system can receive an 8-bits data word from the connected CPU. When receiving such an 8-bit data word, the interface enters an active state, controlling the calculator.
During an active state, a BUSY signal is set by the interface. The BUSY signal can be read from an IN-port in the processor system, or it can be connected as an interrupt signal to the controlling system. In this (first) implementation the BUSY signal is connected as an IN port.

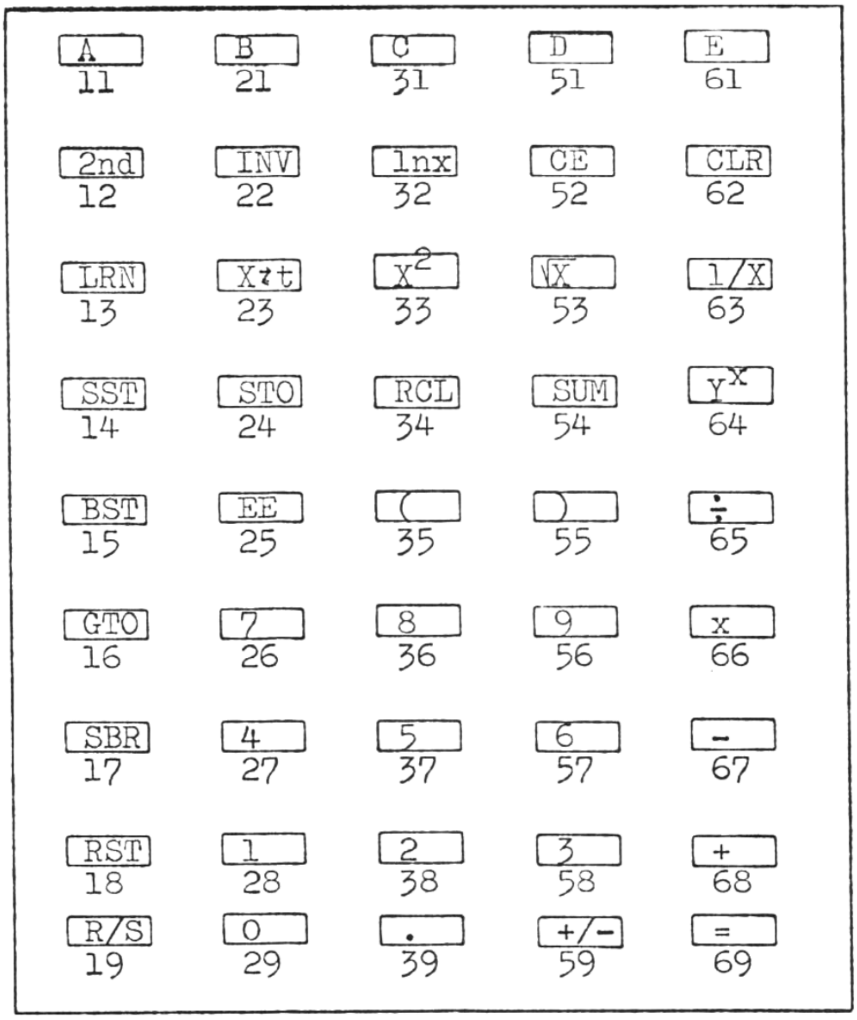
A simulated key activation of the calculator is performed by sending the numeric number of the key in question to the interface (through an OUT-port). The connected calculator will accordingly execute the function of that key, exactly as if it had been depressed on the calculator.

The numeric codes for the keys are not all identical to the numbers used by the calculator for program storeing (the matrix system). Figure 1 shows the hexadecimal codes for the key entry in this interface system.

It should be noted, that the least significant part of the keycode corresponds to the row-number of the key.

Because the simulation of the key-depression is exactly the same as manual operation of the calculator, all activation from the interface must be made as if they were keyed in form the keyboard. This should be noted specially with respect to merged codes.

(F.ex. INV DSZ *12 *15 is keyed in by entering: 22 12 29 12 64 28 38 12 64 28 37 with respect to the codes shown in Figure 1).



A 11	B 21	C 31	D 51	E 61
2nd 12	INV 22	lnx 32	CE 52	CLR 62
LRN 13	X⇄t 23	X^2 33	√X 53	1/X 63
SST 14	STO 24	RCL 34	SUM 54	Y^X 64
BST 15	EE 25	(35	) 55	÷ 65
GTO 16	7 26	8 36	9 56	x 66
SBR 17	4 27	5 37	6 57	− 67
RST 18	1 28	2 38	3 58	+ 68
R/S 19	0 29	. 39	+/− 59	= 69

Figure 1: Key codes

The following example shows a typical key-in sequence:

The functions to be executed by the calculator are:

Value 3.4 is stored in register 05;
The partitioning is set to 799.19 (2 OP 17);
The following program is entered:

LBL **A** **X^2** RET

After this the calculator must enter RUN-mode.

By using the these codes, the described functions are executed by sending the following codes to the interface:

CODE	FUNCTION	
62	CLEAR	
58	3	
39	,	
27	4	
24	STORE	
29	0	
37	5	
38	2	
12	2nd	
56	9	(OP)
28	1	
26	7	
13	LRN	(enter learn mode)
12	2nd	
17	SBR	(LBL)
11	A	
33	x^2	
22	INV	
17	SBR	(RET)
13	LRN	(enter RUN-mode)

This example shows that all merged program codes and operations must be entered slavish as from the calculator keyboard. In the section "User description" further details on program codes and error checks are given.

Interface communication

All codes are send to the interface through an OUT-port, in a 8080 system by the instruction:

OUT nn

where nn is the number of the port, given by the selector switch in the I/O logic. The system status word "C" the "BUSY" signal) can be connected to an IN-port on the same (nn) address.

As long as the "BUSY" signal is "0" it is recommended to send a code to the OUT-port. As long as "BUSY" = "1" the contents of the OUT-port must remain unchanged until "BUSY" = "0".
The synchronization check bit can be read when the "BUSY" line is "0". For proper synchronization the "SYNK OK" bit must be "0" while BUSY is "0". If the calculator and

the interface is not synchronized a new start-up procedure must be performed (see section "Hardware description").

The time necessary to execute a instruction is depending on how long it will take the calculator to perform it. The time can vary from few mS up to many seconds. If the calculator is programmed in LRN-mode the meantime for a total programming of all 960 program steps is approx. 90 seconds, however the speed is depending on the structure of the program (i.e. the number of merged operations).

Hardware description

Firmware in TI-59

All functions in the calculator exists in a so called SCOM (Scanning read Only Memory). This ROM memory contains all the programs the calculator executes - when it is operating - in micro program form (i.e. instruction words to control the logic function of the calculator).

Prior to constructing this interface system, it was examined how the calculator operates when a key on the keyboard is depressed. By means of a logic-analyzer a series of micro instructions was analyzed. Especially the instructions initiated by a key activation has been inspected. Figure 2 shows a printout from the logic-analyzer.

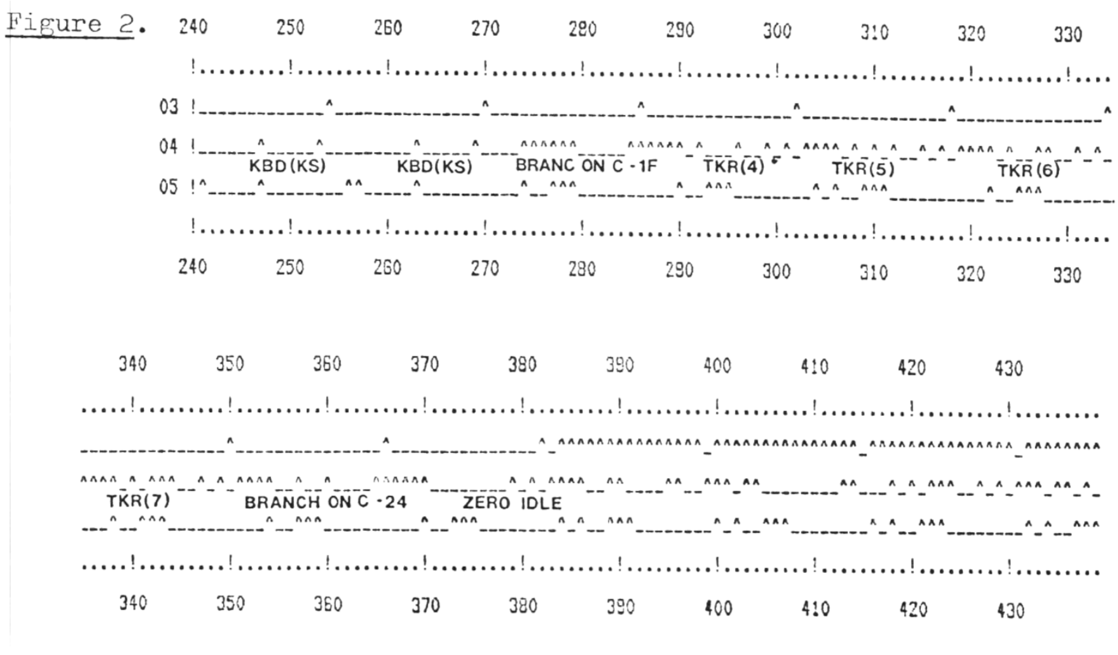


Figure 2: Logic analyzer dump

The three channels used are:

> 03: the IDLE signal in the calculator. Idle is a synchronization signal indicating one instruction period of 16 clockcycles

04: shows the IRG line, a signal line that transports micro-instructions from the SC0Ms to all other devises in the calculator system. The functions of the IRG instructions are noted on the figure, and is explained in details in the following.

05: shows the EXT line, a status- and data line used in the calculator to transport status- and data information between different devises in the system.

The keyboard in the calculator is scanned by the KBD(KS) instruction. The scanning starts in state D15 and continues through D14, D13 ... to D0, after which a hardware construction in the calculators processor circuit makes a test, to see if any key has been depressed. The test executes a conditional (relative) branch. If a key has been activated a COND (condition) latch in the processor will be set, and this is shown by putting the C-bit in the EXT line signal low ("0").
By branching on "if COND = "1" goto -1F" the scanning cycle will continue as long as no key is activated. (Note, the relative goto -1F corresponds to a loop 31 memory steps backwards in the microcode of the calculator).

If COND is "0" the system will start to execute the microcode corresponding to a key activation.

The calculator microcode makes some extra checks on the keyboard register (in the following called KR). Bit 2 to 5 in KR are tested by the operations TRK(4) (Test Keyboard Register bit 4 (..2)). If non of this bits are set, some error condition must have occurred (non of the keys have a code beginning with 0 or ending with 0). If this is the case, the execution is stopped and a branch back to the start of the scanning cycle is made. If the contents of the KR is all right, the IDLE flip-flop is reset (which will increase the clock frequency in the calculator) and the calculator executes the micro instruction program connected to the key in process. It can be noted how IDLE shifts from DISPLAY- to CALCULATE mode (IDLE is phase shifted).

The main idea behind this interface system is, to interrupt the micro program execution in such a way, that the calculator system can be tricked to act as if a key had been depressed. The interruption of the micro program cycle should occur exactly when the "BRANCH 0N C -1F" is executed.

If this BRANCH micro instruction is "destroyed" and overwritten by a"FILL KR" instruction, the calculator system will perceive this as a normal key depression, and will consequently start an execution of the micro program connected to the contents of the KR.

The function of the interface is thus:
When a BRANCH 0N C -1F is discovered on the EXT line, the micro program sequence is stopped by by overwriting a "FILL KR" instruction, after which the BRANCH instruction is destroyed by executing e BRANCH ON C 00 and the control is given back to the calculator without any further interruptions.

Interface

The following description of the whole interface system is divided into 8 parts, each describing the hardware functions in the system. The 8 parts are:

1. Input circuits for signal conversion
2. Synchronization decoding
3. Sequence control by means of ROM memory and counters
4. Output buffers for signal impress in the calculator
5. EXT signal from shift register
6. Calculator state decoding
7. Decoding of BRANCH cycle
8. Other logic for MPU connection (8080 I/O subsystem)

The function of each part, and its place in the interfacesystem is explained.

Input circuits

The interface system is build TTL logic. The Calculator is made in PMOS technique with logic states in the voltage interval O to -16 V, hence a signal level conversion is needed. The conversion is achieved by connecting the ground (Vss) in the calculator to an external +10V power supply (+10V with respect to the TTL system ground). As CMOS buffer circuit CD4050 (IC1) is used for for input conversion from PMOS.

The signals from the calculator are connected to the inputs of the buffers via 10 kOhm resistors.

The signals to be used in the interface are:

- The two clocksignals Ø1 and Ø2
- the IDLE sync. signal and
- the IRG line signal. (See also figure 5 for calculator connections).

Synchronization decoding

The negative going edge of the IDLE signal indicates the start of the S0 state of the calculator. Many functions in the interface need a short puls at state S0. Such a puls is genrated in a mono stable latch IC3 (74LS221).

The time constant is very short - approx. 50 nS. The latch is triggered at the negative going edge of IDLE and so the internal generated IDLEX signal can be used directly to control reset and other interface functions.

ROM control

The interface is build as a sequence machine with its controlling instructions stored in a ROM (IC11).

A ROM type 2700 is used, but any memory circuit with 8 bits output and a minimum of 64 words can be used. (In later versions the 2700 UVPROM was replaces by an EEPROM type 28C64).

The addresses for the ROM are generated in two synchronous counters IC8 and IC9. Thees counters are clocked with Ø1, and are in this way working in parallel with the state decoder in
the calculator system. Each of the 8 bits from the ROM have separate functions.

These functions are:

BIT 0: IRGKODE used to generate codes for the IRG line to the calculator. The sending of data on the IRG line is controlled by the IRGEN signal.

BIT 1: IRGEN. This bit controls the output buffer circuit for sending data on the IRG line. If the IRGEN = "0" the buffer is open for transmission of IRGKODE data to the IRG line.

BIT 2: EXTKODE. If a specific code is to be send to the EXT line, it can be stored as bit 2. It is possible to select the source of the EXT code. Either the code comes form the ROM or form a shift register.

BIT 3: SHIFT/CODE determines the source of the EXT code. Bit 3 = "1" enables userdata from the shift register (explained later), and bit 3 = "0" enables EXT data from bit 2 of the ROM.

BIT 4: EXTEN, when "0" the EXT line buffer is enabled, allowing data from the interface to be send to the calculator.

BIT 5: EXTLOAD controls the loading of the shift register for EXT codes. Because the shift register is clocked by Ø1 , the load signal must appear 1 clock state before the sending of the EXT code form the shift register is started.

BIT 6: STOP controls the counters so that a STOP bit can stop the ROM counters and make the interface system enter a "SYNC OK" state.

BIT 7: CIRCULATE makes sure that the counters are enabled when a sending sequence is started.

Because the ROM counters are clocked on Ø1, a new word from the ROM is read out at every calculator state. This makes it very easy to store IRG and EXT instructions in the ROM.

BIT 6 and BIT 7 can only be programmed in one way. The STOP bit must always be at "1" except in the ROM word following the last code word in the ROM. BIT 7 must always be "0" except for the first and the last ROM word. The two line control signals (the IRGEN and EXTEN) must always be disabled in the first and the last ROM word.

Sending of code is initiated by the MPU system by setting the latch IC13 (74LS74). By this, a load signal through IC13 is submitted to the ROM counters. All preset-inputs to this counters are at "0". The load signal forces the counters to send address 00 to the ROM. In this state the system is in a idle-mode, awaiting a start signal to begin code sending.

The start signal occurs when the IRG instruction BRANCH ON C -1F is decoded in the IRG decoding logic. The start signal enables the counter-clock inputs, and the ROM codes
are now controlling the EXT and the IRG lines in the calculator.

Output buffers

To impose signals on to the EXT and the IRG lines in the calculator, special buffer circuits are necessary. The buffer consists of two transistors and two TTL tri-state buffers IC18 and IC20 (74LS125). By controlling the two transistors form the TTL buffers it is possible to:

1. convert the TTL logic level to the PMOS level in the calculator, as well as generating the necessary current to overwrite the signals on the calculator lines, and
2. disable the buffer circuit by disabling the TTL buffers, so that the ON/OFF control of the line buffers becomes very simple.

The buffer stage is works inverted in the calculator, so that a logic "1" from the TTL logic is perceived as logical "0" by the calculator. It is important to supply the calculator with the correct logical values, observing that inverse IRG codes are send from the ROM and that inverse EXT codes are send from the selector logic.

EXT code selection

From the MPU system a 8-bit data word can be written into a 8-bit wide shift register IC7 (74LS165). The loading of this register is controlled by BIT 5 in the ROM word. The output from this shift register is connected to a selector logic consisting of IC2 and IC1O. The selection of the EXT code is controlled by BIT 3 in the ROM word. The output form the two NAND gates are OR'ed and the output is connected to the EXT line buffer logic.

Data form the shift register is clocked by Ø1. The calculator system reads the IRG and EXT codes on the negative going edge of Ø2. Because the ROM counters are working synchronous with the calculator statecounter the load signal to the shift register must be generated one clock cycle before the first bit in the register is to be used as a EXT data bit.

State decoding

The internal S states of the calculator are decoded by the counter IC4 (74 LS161). The counter is clocked by Ø1 and it is reset by IDLEX. The output from the counter is thus representing the actual calculator state. From the state decoding a EOC (End Of Cycle) signal is generated.
This signal is "1" when the calculator is in display-mode (i.e. the internal IDEL latch is set, slow clock), and "0" when the calculator is in calculate-mode (IDLE latch reset, fast clock). The EOC signal is generated by "reading" IDEL over Ø2 (this is done in IC5) and then use this signal to clock the decoded S1 state. If IDLE indicates display-mode a "1" will be clocked into the latch, otherwise a "0" is clocked in.
The EOC signal is used to control the BUSY latch IC13, and together with the decoded IRG line signal to enable the ROM counters.

IRG instruction decoding

The decoding of the IRG instructions are done successively bit by bit in each calculator state. The bit sequence to decode is the binary representation og the BRANCH instruction and is set up on the inputs of a 16 to 1 decoder IC21 (74150),which is supplied with the calculator state counter output from IC4. The output from the decoder is directly compared to the IRG line signal in a EXCLUSVIE

OR gate IC14. The result of the comparison is clocked into a JK latch on Ø2. If all bits on the IRG line matches the bit combination om IC21 then a "0" is clocked into the JK latch.

If there is no match, then a "1" is clocked in, and this "bit-error" stays here until it is clocked into the next JK latch. (Bit error here means, a no match, i.e. we are still waiting for the BRANCHE command to appear on the IRG line).

The first JK latch is reset by IDLEX after a complete instruction cycle. The second JK latch is clocked by the ripple carry clock from the state counter IC4. The ripple clock is active in state S15. This means that only the 15 first bits (from calculator state S0 to S14) are included in the comparison. The last bit in the IRG words indicates to the calculator system , if a unconditional un-relative branch is to be executed. Because all conditional branches all have three "0" as first bits
it is not necessary to include the last state 15 bit in the decoding of the branch instruction. If the bit comparison shows consistency the Q output from the second JK latch is "1". If the calculator is still in display mode (with EOC="1") the START/STOP latch IC6 input is supplied with a "0". At the next IDLEX (the next S0 state) this "0" is clocked into the START/S TOP latch and if the ROM counters are loaded with address 00 (i.e. that the first ROM word is on the output of the ROM) a code sending sequence is started. If the ROM is not at address 00, the system is waiting for a load puls from the MPU, and a new branch instruction must be decoded before a code sending can be executed.

<u>I/O logic for MPU connection</u>

The constructed I/O logic is developed for a 8080A micro processor system. For other MPU systems the I/O subsystem can be modified so that it will fit the desired processor system. The following property of the I/O subsystem should be noted:

The word read by a IN nn instruction (nn is the number of the port selected by the switch), only contains information in the two least significant bits. BIT 0 is the BUSY signal from IC13 and BIT 1 is the STOP signal from the ROM. The STOP signal indicates if the calculator is synchronized with the interface, because the ROM counters, if no error states has occurred, always stops when STOP ="0".

Main logic functions

The main logical functions are system functions that are not directly given by the hardware of the system. The functions in question are the sequence of IRG and EXT codes in the. ROM and the selection of the "trigger word" from the IRG line.

Decoding of the trigger word

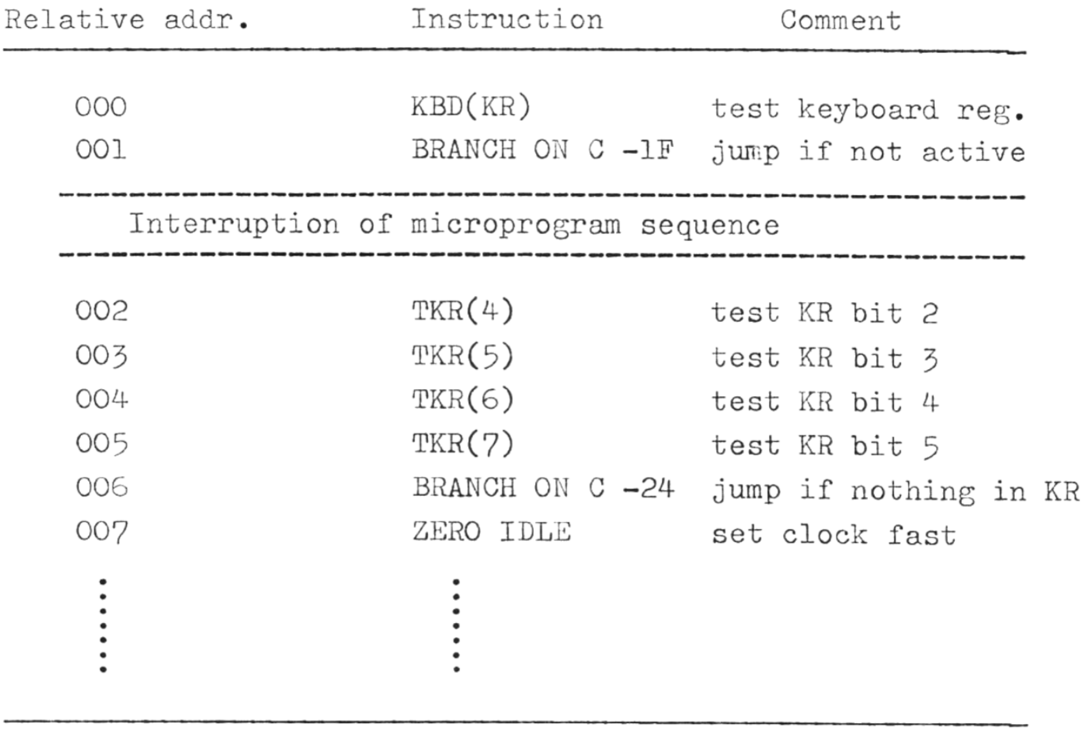
As described, the micro prograrn execution in the calculator must be interrupted immediately after a complete keyboard scanning, where the conditional branch tests the KR. The binary value of the branch instruction can be found in the logic analyzer printout (Figure 2), and for TI-59 and TI-58 calculators it is:

```
S0                    S15
  0001111110000011      = 0xC1F8
```

This means that to decode this branch instruction, the value 0xC1F8 must be set on the decoder circuit IC21. It should be noted, that for TI-58C calculator, the branch at this point is made to another location. To use the system with a TI-58C it is necessary to change the trigger word. The TI-58C trigger word is not available at this point.

The ROM sequence execution

During execution of micro instructions in the calculator, an internal micro program counter is counted up. When a micro instruction is read from a specific address, the contents of the micro program counter is adjusted depending of the three control bits transmitted in the following instruction cycle (on the EXT line). This means, that the contents of the micro program counter remains unchanged until state 23 in the following instruction cycle. This fact is very important with respect to the possibilities of sending 'false' signals to the calculator. Thus it is utilized that the micro program counter can be stopped by setting the HOLD bit in the EXT control word. When a branch instruction is decoded from the IRG line, the interface system starts by sending "false" codes in the following instruction cycle, in witch the HOLD bit is set. In the succeeding cycles the HOLD bit remains set, so that the program counter, after the HOLD bit has been resat, will point to the micro instruction immediatly after the branch instruction. shows how the micro-processor instructions in the calculator system are build up and how the "false" instructions are placed in the micro program sequence.

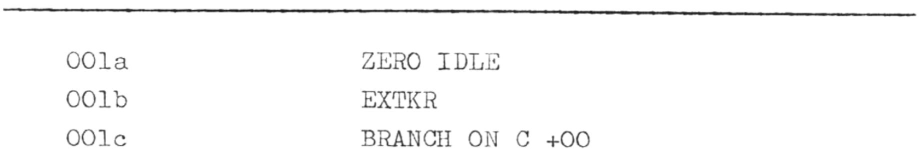


```
Relative addr.          Instruction         Comment
------------------------------------------------------------

    000                 KBD(KR)             test keyboard reg.
    001                 BRANCH ON C -1F     jump if not active
    --------------------------------------------------------
       Interruption of microprogram sequence
    --------------------------------------------------------

    002                 TKR(4)              test KR bit 2
    003                 TKR(5)              test KR bit 3
    004                 TKR(6)              test KR bit 4
    005                 TKR(7)              test KR bit 5
    006                 BRANCH ON C -24     jump if nothing in KR
    007                 ZERO IDLE           set clock fast
     .                    .
     .                    .
     .                    .
     .                    .
     .                    .
     .                    .
     .                    .
------------------------------------------------------------
```

While the micro program execution is stopped (after decoding of the BRANCH ON C - 1F) three "false" micro instructions are executed. The purpose of thees instructions are partly to enter a keyboard value in the keyboard register (KR) and partly to overwrite the branch instruction. The three micro instructions entered on the IRG line are:

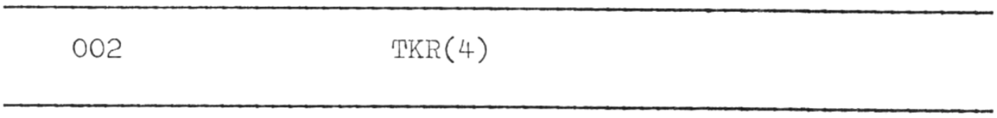


```
001a                ZERO IDLE
001b                EXTKR
001c                BRANCH ON C +00
```

The function of each instruction is:

1.a Set the clock frequency fast and force the calculator system into calculate mode . At the same time the HOLD bit on the EXT line is set.
1.b Fill the KR with the value transmitted on the EXT line, and keep the HOLD bit set.
1.c Destroy the old branch instruction by jumping nowhere (+00 relative to the current contents of the micro program counter).

After this both line transmitters are disabled, and the calculator system now executes the instruction exactly as if a key had been depressed on the keyboard. The value entered into KR determines witch key-function the calculator then will execute.



```
002                 TKR(4)
```

Figure 3b: Next executed instruction after release

User instruction

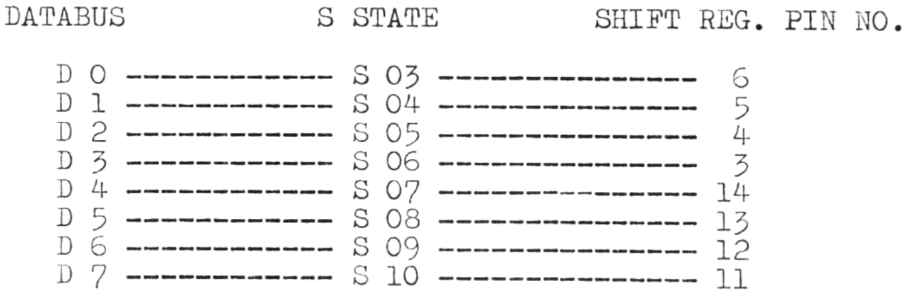
To make the user instruction more clear it is divided into three parts covering:

1. The logical connection between MPU and interface.
2. Code sending. How to enter keycodes for proper calculator operation.
3. Startup of the system.

Logical MPU connection

Figure 4 shows an example of an I/O subsystem developed for a 8080A processor system. There are two fundamentally ways to connect a I/O subsystem to the interface.

The databus (8-bit wide) can be connected directly to the shift register IC7.



DATABUS	S STATE	SHIFT REG. PIN NO.
D 0	S 03	6
D 1	S 04	5
D 2	S 05	4
D 3	S 06	3
D 4	S 07	14
D 5	S 08	13
D 6	S 09	12
D 7	S 10	11

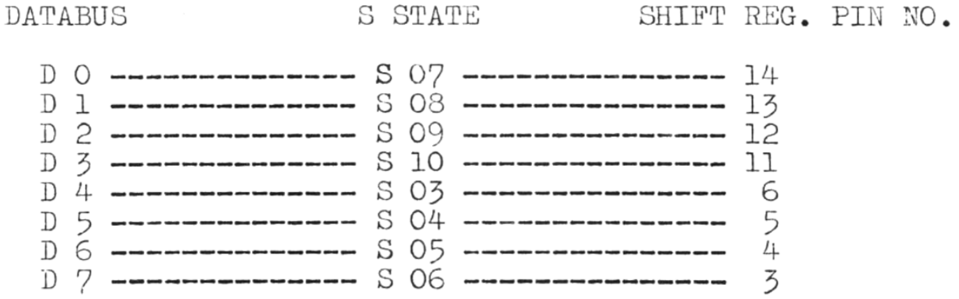
Figur 4: Connect databus directly to shift register IC7

Note that the S state indicates at what calculator sequence state the individual bits are transmitted to the EXT line.

The shown connection implies, that the dataword send to the shift register has exactly the same format as the KR register contents when a key is depressed. The hexadecimal values shown in figure 1 and in table 2 is based on this form of connection. The method has a disadvantage because the codes are representing the keys in reverse order with respect to normal matrix notation.

Example: The C key has in ordinary notation the code 13 but the value 31 must be send to the interface system because the KR register in the internal calculator system is read backwards.

This inconvenience can be avoided, so that the key codes can be transferred in normal TI-59 format.This is done by exchanching the four least significant bits in the dataword with the four most significant bits. A connection like that implies that the row number becomes the first part of the data word, and the column number becomes the last part, however the column number will be out of order for column 4 and 5 (represented by the values 5 and 6). The schematic connection of the latter connection method is shown below:



DATABUS	S STATE	SHIFT REG. PIN NO.
D 0	S 07	14
D 1	S 08	13
D 2	S 09	12
D 3	S 10	11
D 4	S 03	6
D 5	S 04	5
D 6	S 05	4
D 7	S 06	3

Figur 4a: Revers order interface connection

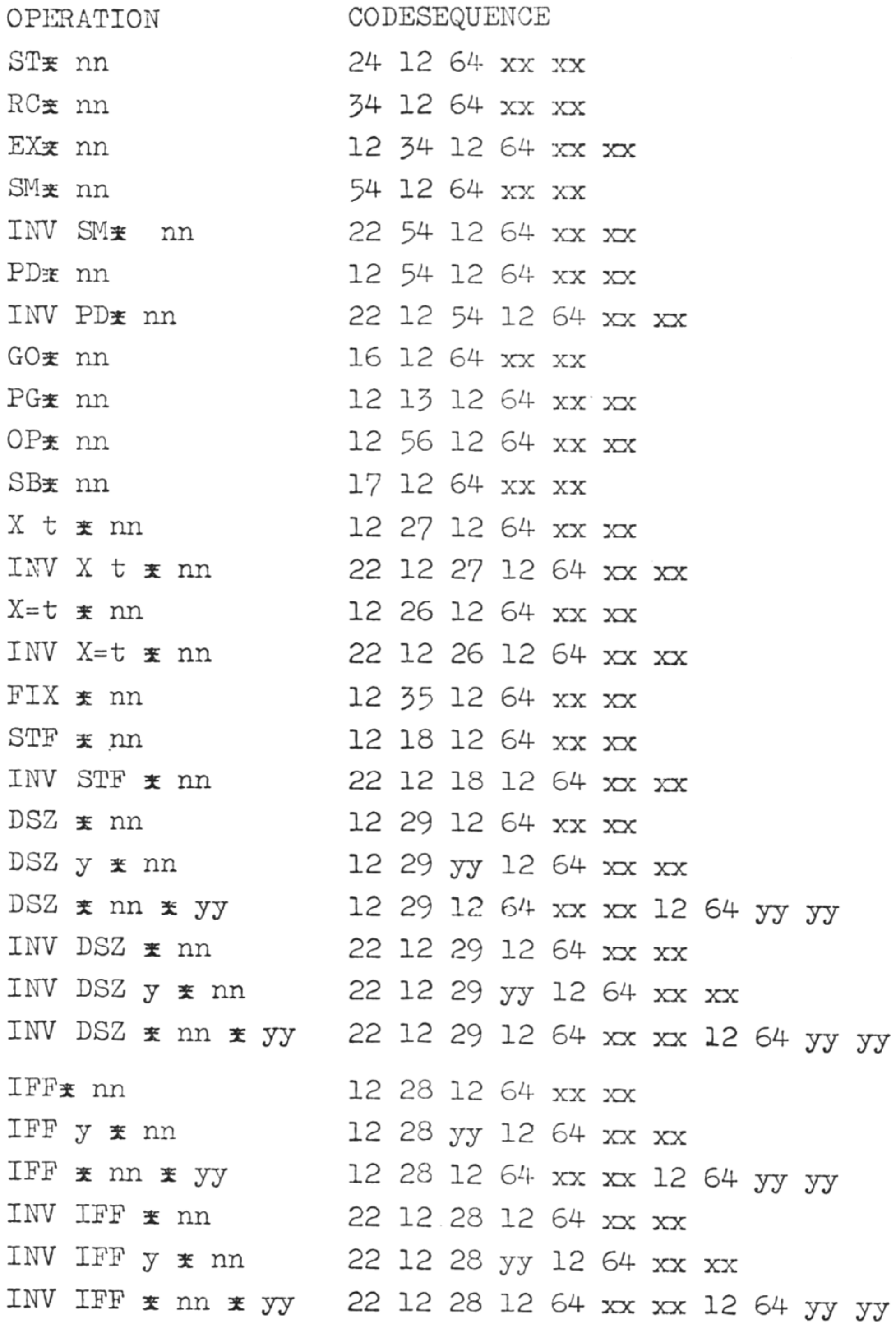
Code sending and input check

As described in the introduction, it is fairly easy to send codes to the calculator through the interface system. However some error-situations may occure if the output data words are not checked before they are send to the interface. The errors will occure in the calculator, if the keyboard register is filled with an "illegal" keyboard code. An "illegal" code is a code containing a non existent keyboard column number (column 0, 4 and column numbers greater than 6 does not exist) or a nonexistent keyboard row number (row numbers greater than 9 or equal to 0).

The micro program in the calculator is not able to check if the contents of the *KR* is legal, and the calculator will perform nonsense operations if the KR register holds a wrong code. In most error situations the calculator will lock up in an endless loop in calculate mode. That means that the EOC logic in the interface never recieves a "ready" signal, and hence further code sending is not possible.

The only way to get out of such a loop is to turn off the calculator and then go through a new startup procedure . All programs controlling the interface should therefore implement suitable error-check routines to catch wrong key codes.

To facilitate the entry of merged codes the table below shows how all forms of merged codes can be programmed. The values in the table is based on direct MPU/shift register connection (i.e. databit 0 in S03 to pin no. 6 and D7 in S10 to pin no. 11).



OPERATION	CODESEQUENCE
ST* nn	24 12 64 xx xx
RC* nn	34 12 64 xx xx
EX* nn	12 34 12 64 xx xx
SM* nn	54 12 64 xx xx
INV SM* nn	22 54 12 64 xx xx
PD* nn	12 54 12 64 xx xx
INV PD* nn	22 12 54 12 64 xx xx
GO* nn	16 12 64 xx xx
PG* nn	12 13 12 64 xx xx
OP* nn	12 56 12 64 xx xx
SB* nn	17 12 64 xx xx
X t * nn	12 27 12 64 xx xx
INV X t * nn	22 12 27 12 64 xx xx
X=t * nn	12 26 12 64 xx xx
INV X=t * nn	22 12 26 12 64 xx xx
FIX * nn	12 35 12 64 xx xx
STF * nn	12 18 12 64 xx xx
INV STF * nn	22 12 18 12 64 xx xx
DSZ * nn	12 29 12 64 xx xx
DSZ y * nn	12 29 yy 12 64 xx xx
DSZ * nn * yy	12 29 12 64 xx xx 12 64 yy yy
INV DSZ * nn	22 12 29 12 64 xx xx
INV DSZ y * nn	22 12 29 yy 12 64 xx xx
INV DSZ * nn * yy	22 12 29 12 64 xx xx 12 64 yy yy
IFF* nn	12 28 12 64 xx xx
IFF y * nn	12 28 yy 12 64 xx xx
IFF * nn * yy	12 28 12 64 xx xx 12 64 yy yy
INV IFF * nn	22 12 28 12 64 xx xx
INV IFF y * nn	22 12 28 yy 12 64 xx xx
INV IFF * nn * yy	22 12 28 12 64 xx xx 12 64 yy yy

System startup

The startup procedure is in many ways dependent on the properties of the MPU system. The following general points show how a typical startup may take place:

1. Turn on the MPU system and all equipment connected to the processor, including the +5V power supply for the interface system

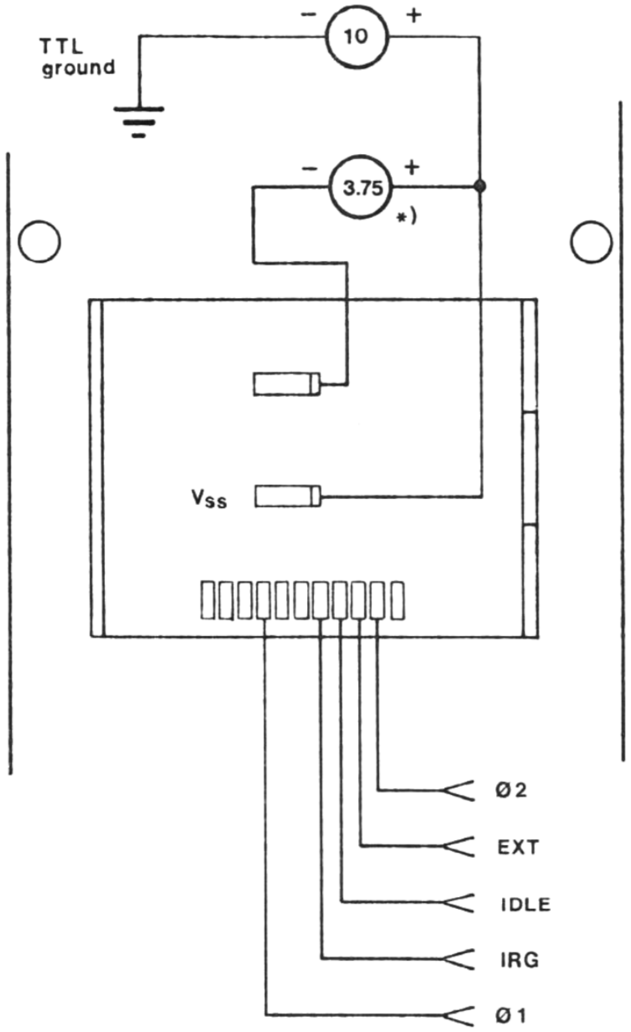
2. Connect the interface system to the calculator.

3. Turn on the +10V power supply to the interface.
4. Now the calculator can be turned on. If the calculator enters calculate mode (the display turns off) it can probably be brought back to display mode by pressing the R/S k ey. If not, turn the calculator off and on again. It should be noted that it is the internal calculator clock that controllers the interface system, and that this point in the startup procedure is not depending on the MPU system.

It should also be noted that the +I10V power supply always must be on when the calculator is turned on. It will not damage the calculator if not, but it may be difficult to get it into display mode .
When the calculator is under control of the interface system it is a good rule not to touch any key on the keyboard.

If a key is depressed, the interface will stop further code sending until the key is released, however the key activation may disturb the sequence of codes from the controlling system.

Calculator connection

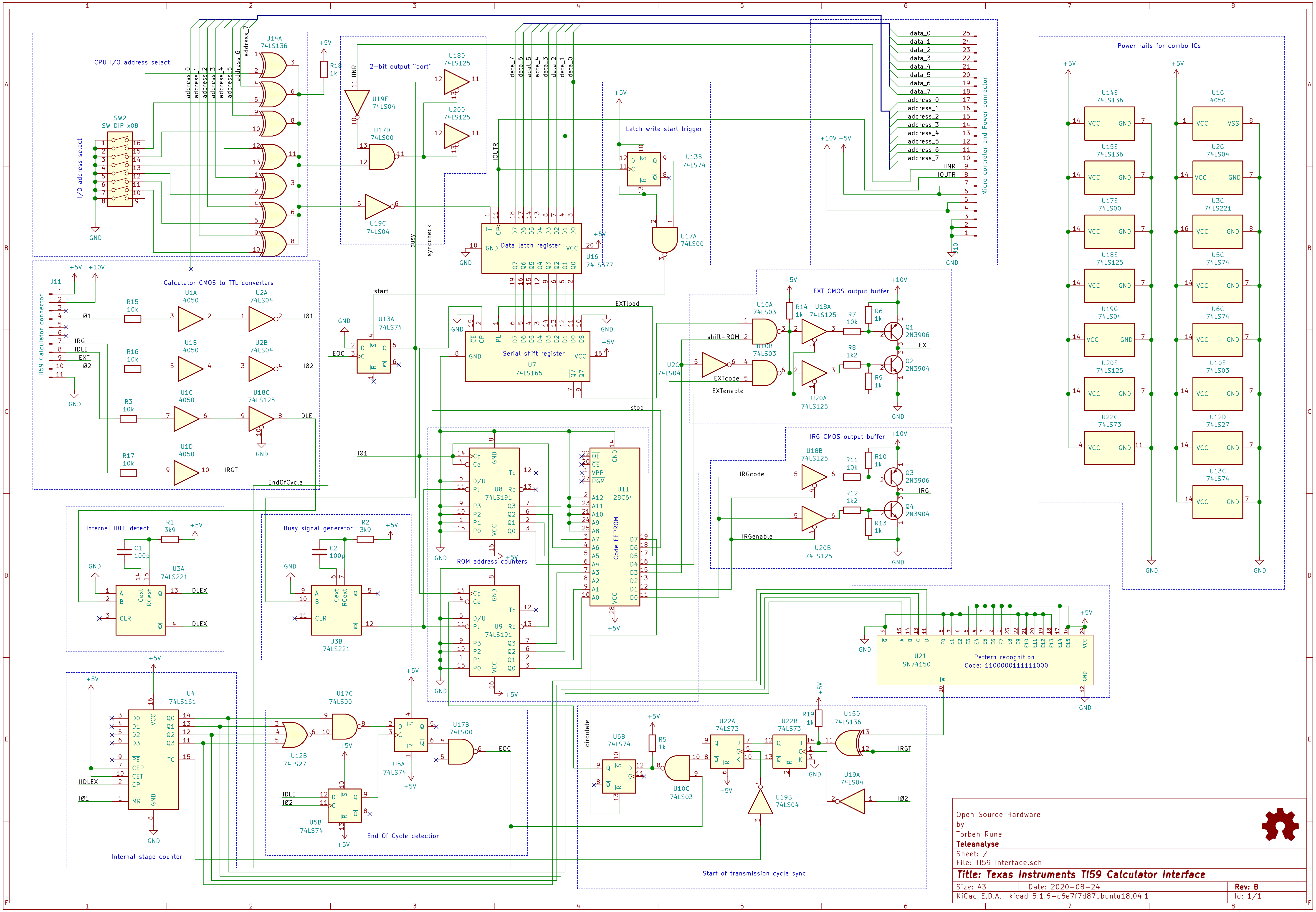


This figure shows how to connect the calculator with the interface system. The calculator is shown form the back with the battery pack removed. The lowest row of connection points are connected to the respective interface lines. The V_{ss} ground terminal in the calculator is connected to a 10V power supply.

*) The calculator can be supplied in several ways, for instance by:

- a an external 3.75 V power supply
- b the removed battery pack
- c if used on a PC-100C printer, the internal printer supply.

If a printer is used, the connection to the connection points becomes very easy. Wires can simply be soldered to the solder points in the printer.



Open Source Hardware
by
Torben Rune
Teleanalyse

Sheet: /
File: TI59 Interface.sch

Title: Texas Instruments TI59 Calculator Interface

Size: A3 Date: 2020-08-24 **Rev: B**
KiCad E.D.A. kicad 5.1.6-c6e7f7d87ubuntu18.04.1 Id: 1/1

Bill of Materials

Id	Designator	Package	Quantity	Designation
1	J10	PinHeader_1x25_P2.54mm_Horizontal	1	Micro controler and Power connector
2		MountingHole_3.2mm_M3	4	MountingHole_3.2mm_M3
3	Logo	TELEANALYSE	1	LOGO
4	Open HW logo	OSHW-Symbol_6.7x6mm_SilkScreen	1	OSHW-Symbol_6.7x6mm_SilkScreen
5	R19,R5,R6,R9,R10,R13,R14,R18	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	8	1k
6	R1,R2	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	2	3k9
7	U22	DIP-14_W7.62mm	1	74LS73
8	U12	DIP-14_W7.62mm	1	74LS27
9	U13,U5,U6	DIP-14_W7.62mm	3	74LS74
10	U21	DIP-24_W15.24mm	1	SN74150
11	C1,C2	C_Axial_L3.8mm_D2.6mm_P7.50mm_Horizontal	2	100p
12	Q1,Q3	TO-92_Inline	2	2N3906
13	Q2,Q4	TO-92_Inline	2	2N3904
14	R3,R7,R11,R15,R16,R17	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	6	10k
15	R8,R12	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	2	1k2
16	SW2	SW_DIP_SPSTx08_Slide_6.7x21.88mm_W7.62mm_P2.54mm_LowProfile	1	SW_DIP_x08
17	U1	DIP-16_W7.62mm	1	4050
18	U2,U19	DIP-14_W7.62mm	2	74LS04
19	U3	DIP-16_W7.62mm	1	74LS221
20	U4	DIP-16_W7.62mm	1	74LS161
21	U7	DIP-16_W7.62mm	1	74LS165
22	U8,U9	DIP-16_W7.62mm	2	74LS191
23	U10	DIP-14_W7.62mm	1	74LS03
24	U15,U14	DIP-14_W7.62mm	2	74LS136
25	U17	DIP-14_W7.62mm	1	74LS00
26	U18,U20	DIP-14_W7.62mm	2	74LS125
27	J11	PinHeader_1x11_P2.54mm_Horizontal	1	TI59 Calculator connector
28	U11	DIP-28_W15.24mm_Socket	1	2764
29	U16	DIP-20_W7.62mm	1	74LS377