INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

The United Nations University

SMR/774 - 22

THIRD COLLEGE ON MICROPROCESSOR-BASED REAL-TIME CONTROL - PRINCIPLES AND APPLICATIONS IN PHYSICS
26 September - 21 October 1994

A SIMPLE DEVICE FOR LINUX: ICTP & GPI BOARDS

Ravindra KARNAD
Five-D Electronic Technical Services
No. 25 Anniyappa Garden
6th Main. 10th Cross
Thippasandra
560 075 Bangalore
INIDA

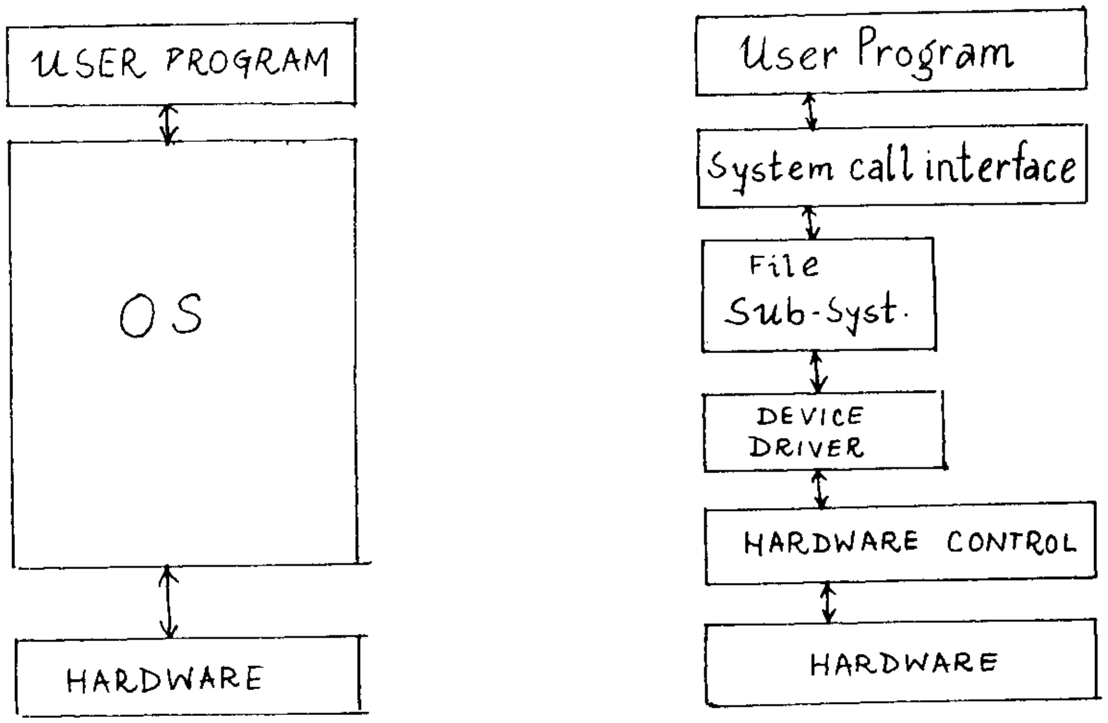
THIRD COLLEGE ON MICROPROCESSOR BASED REAL-TIME CONTROL
TRIESTE 1994

A SIMPLE DEVICE FOR LINUX: ICTP & GPI BOARDS

Ravindra Karnad
INDIA

∩

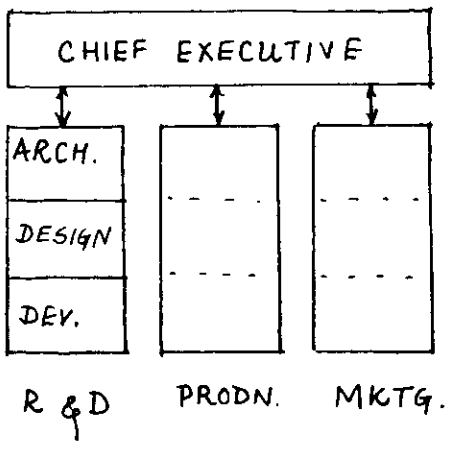
- An operating system hides from the user all the cumbersome and unpleasant details of the hardware.
- An OS (like LINUX) allows the user program in a language like "C" to access any hardware external to the PC like a printer or disk -drive without having to bother about the details of theses devices.
- A structured OS would permit the user to access any hardware just like he would access a file.
- The OS presents a layer which hides the messy hardware and presents a clean interface.
- The OS itself is internally structured (as shown in Fig 1) and has portions of it dedicated to perform I/O. With this layered model, when a user programming in "C" issues an OS call like "read" or "write" for device I/O, such a call percolates down this hierarchy of layers. As the control flows down through these different layers, more tasks/processes are spawned and functions within the kernel are invoked which directly interact with the hardware and also handle interrupts.
- The ICTP & GPI boards provide us with a simple example of a character oriented device which can be interfaced to LINUX through a device driver.
- With the help of these boards and LINUX we can understand how a structured real-time control system might be built and used.



LAYERED MODEL OF THE OS.

TYPICAL STRUCTURE OF A REAL-TIME CONTROL SYSTEM:

The layered structure of a Real-time control system can be understood by drawing a parallel to some other structured systems that we see around us in daily life. Consider for example the structure of a corporate body:

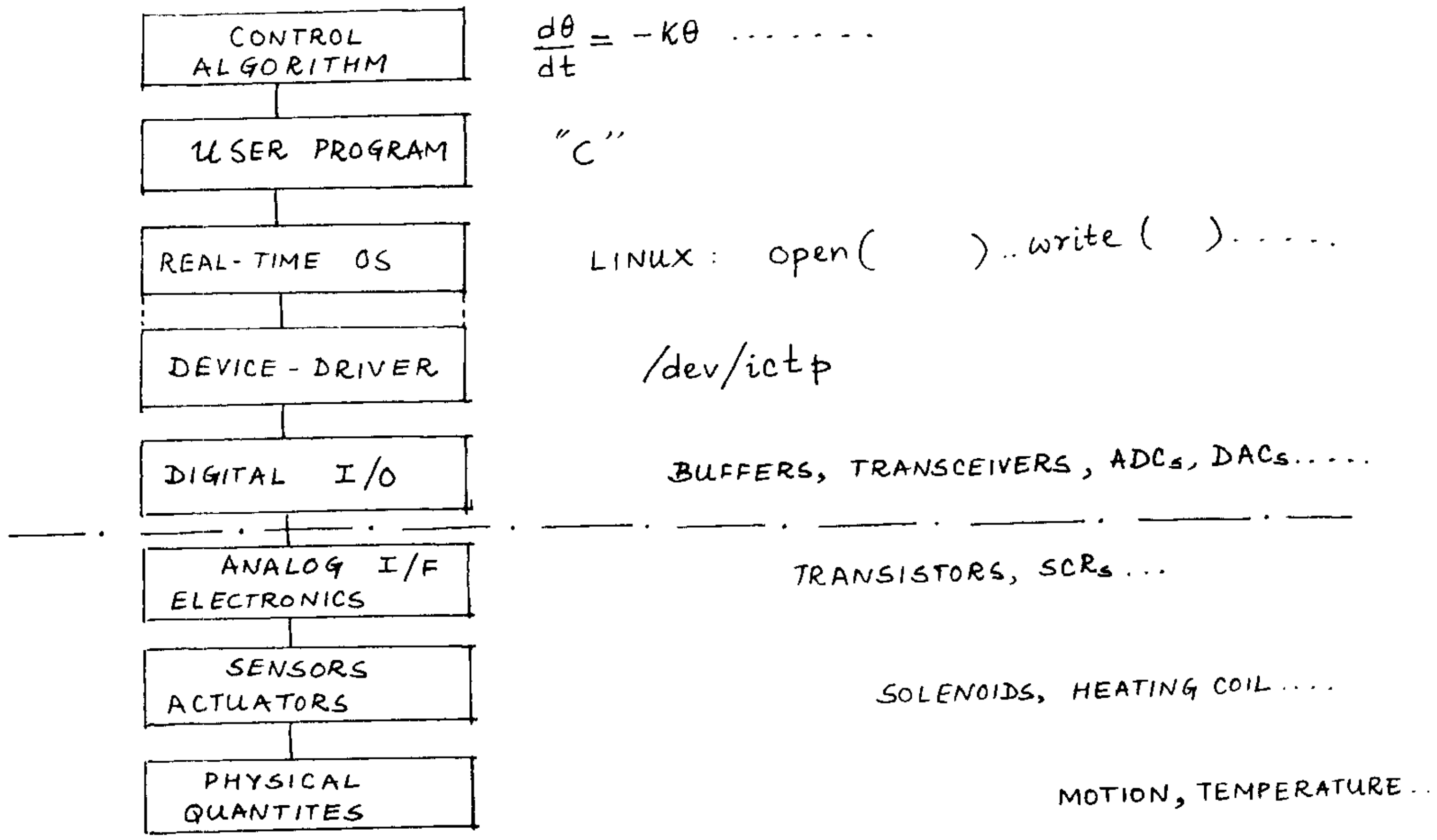


- The idea for a new product percolates down the hierarchy in a top-down fashion.
- Starting at the top there is only a concept or an idea which is gradually refined, developed , and worked upon by the lower layers till the product is finished.

Compare this with the layers of a real-time control system:

- What started as a mere idea at the top begins to spwan or create more work for the lower levels **(child-processes are created)**
- Each layer has a **well defined function.**
- Each layer has a **well-defined interface** to it's neighbouring layer (either the top / bottom)
- **Details** of what goes on in each layer can be (and usually) is **well hidden from the other layers**.

In a real-time control system the layers usually encountered are:



- The **control algorithm** forms the top layer which is then implemented as:
- A **user program** which runs on:
- A **real-time OS** like OS-9 or LINUX (for our purposes) which has a:
- The **device driver** (actually a part of the OS 's kernel) written to handle:
- The **digital interface** controlled by the driver
- **Analog interface electronics** to interface to:
- **Transducers / Sensors /Actuators** which:
- Measure / detect / cause changes in the outside **physical world** such as temperature, motion, Magnetic field etc.

Drawing an analogy to the corporate body structure, we can once again note the similarities and the advantages of following a structured approach:

- ***Within each layer since the job is well defined***, during implementation it enables one to concentrate on the function of that layer without having to bother too much about other layers. For eg the person working on the control algorithm need not bother either about the syntax of "C" or the intricacies of interfacing a particular transducer.
- ***Changes in one layer must ideally be done so as to make them transparent to other layers***. For instance if the electronics associated with a transducer is re-designed, as long it works well over the required range the rest of the system need not know about it. Also if one of the devices is changed completely, the device driver alone would be probably modified.
- Each layer calls for some ***specialised expertise and standards.*** One could use freely available resources rather than resorting to *doing it yourself* all the time.
- It is ***easier to make changes or to duplicate*** the system.

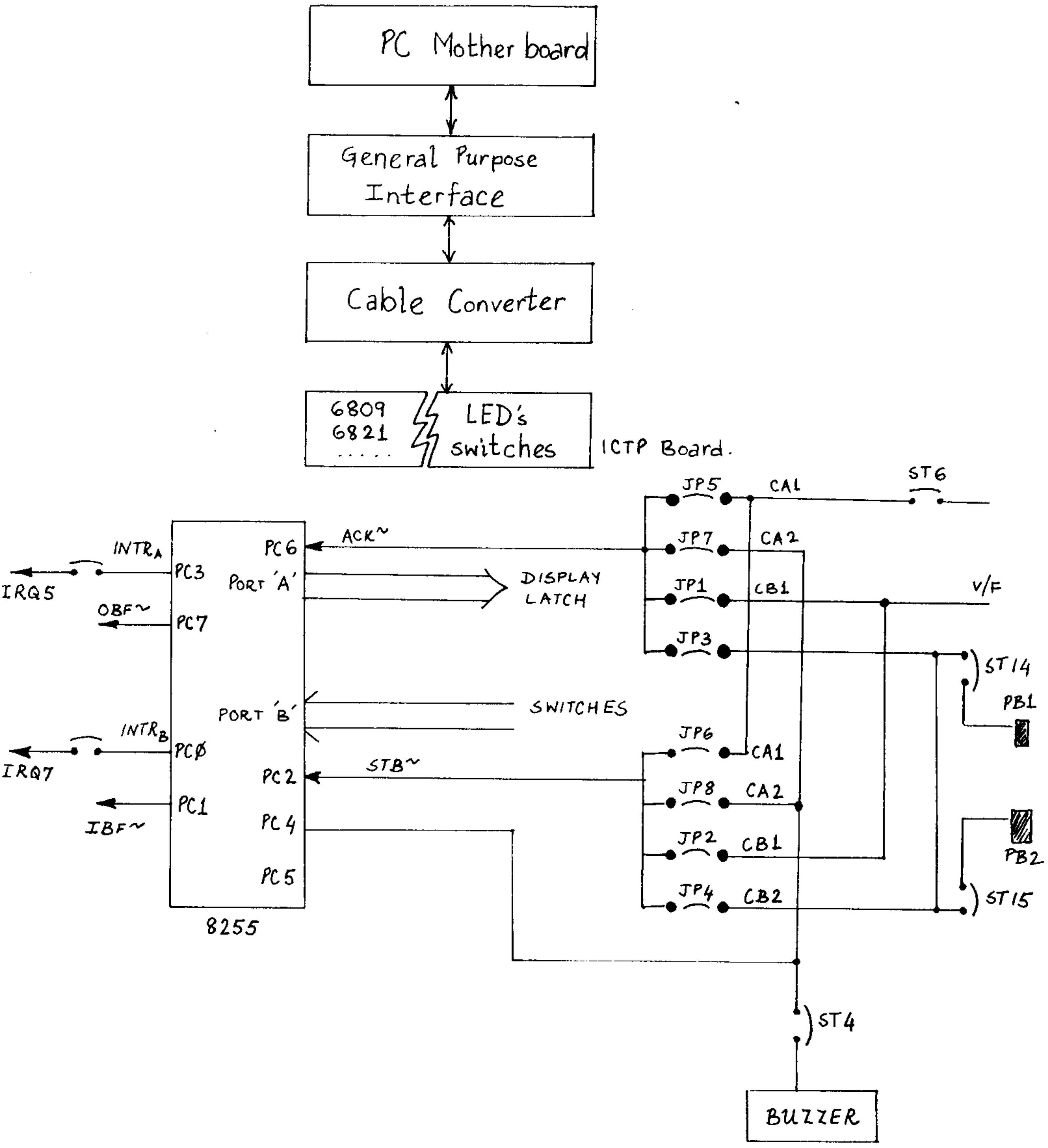
Note however that:

- The separation of functions into layers is largely dependent on the system-designer.
- The dividing line shown in the figure indicates the scope of the programmer's model. Beyond this either the system would be completely hardware with no software or would be implemented with an embedded system.
- The scope of the present course is restricted to the first 4 layers.
- The GPI-ICTP board represents the dividing line.

- The GPI-ICTP boards represent the lowest programmable level.
- Beyond this the inputs come from sensors or detectors through the associated electronics. This is simulated with LEDs and switches. So in short the GPI-ICTP boards along with your "C" program, the device-driver, LINUX and your 486 machine form a **simple representative model of a well-structured real-time system.**

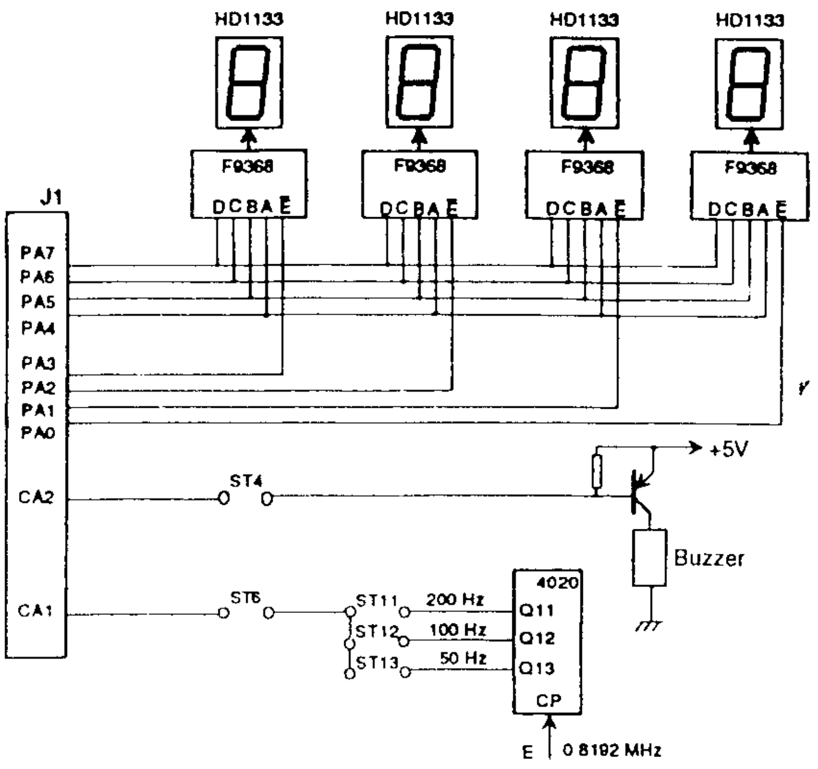
- When working with these boards remember to concentrate on the concepts that are being demonstarted. Do not worry yourself too much about the (apparent) simplicity of the "system" and other minor details of the board.

- Remember that the concepts are yours. Take them home and try them out on your own system and experiments (But the GPI-ICTP boards will be useful only so long as the course lasts!)

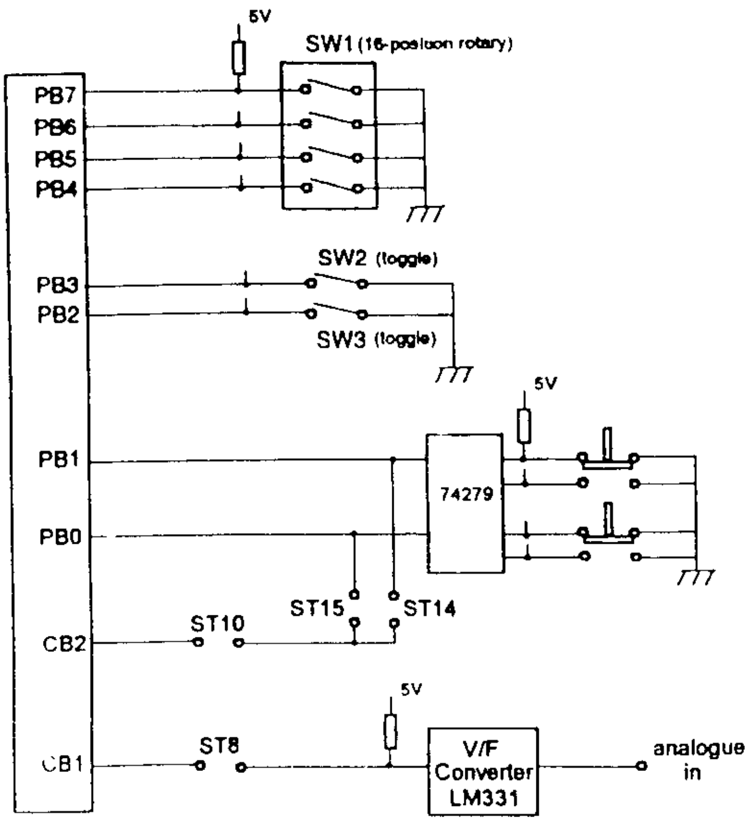


BLOCK DIAGRAM OF GPI - ICTP.

I/O Section Port A



I/O Section Port B


5V
SW1 (16-position rotary)
PB7
PB6
PB5
PB4
SW2 (toggle)
PB3
PB2
SW3 (toggle)
5V
PB1
74279
PB0
ST15
ST14
ST10
CB2
5V
ST8
CB1
V/F Converter LM331
analogue in