UNIVERSAL DIGITAL AGGREGATOR FOR IN-LINE SIGNAL PROCESSING

M. P. Kopeć*, Ł. J. Dudek, A. Kisiel, M. Knafel, A. I. Wawrzyniak, M. Zając.
Solaris National Synchrotron Radiation Centre, Jagiellonian University, Krakow, Poland.

ABSTRACT

Universal digital aggregator is a device for general signal processing with around 100kHz bandwidth. It consists of 4 inputs and 4 open-drain outputs - all of which are fully programmable. When the number of controlling digital signals exceeds the number of input ports of a device there is a need to either multiplex those signals or process them before the target device. The aggregator can be powered from the target device so no additional cabling is needed, especially considering its low power consumption. This straightforward, complex and portable device can be easily applied where PLC solutions are difficult to implement.

INTRODUCTION

The purpose of this device is to convert a set of 4 input signals to 4 outputs taking into account a given algorithm. This is useful whenever connecting to a device that cannot perform necessary data processing. Additionally the device can operate with supply voltages in the range between 2 V and 5.5 V, which most PLC devices exceed. This, together with the low power consumption, makes for a great in-line device, that can operate without plugging it into the mains socket, as long as the target or the source device (for which the signal is processed) can provide power within the acceptable voltage range. If there is no suitable power source coming from the device it is still possible to run the device on a standard power brick connected to the mains, outputting acceptable voltage.

HARDWARE

The Universal Digital Aggregator's hardware was designed to provide maximum flexibility with as few components as possible. The structure of the hardware can be split into 5 main parts:

- Supply voltage selection
- Input voltage filtering
- Input protection
- Output protection
- Microcontroller

The device is shown in Fig. 1.

Supply Voltage Selection

Having in mind, that a variety of voltage sources may be needed, the device is equipped with a selectable power input stage. It can operate without any external power, driven by the input signals from the ICEPAP controller, as well as from a stationary power supply. Power from serial programming port for the microcontroller is also possible. To set any given power mode one needs to select a jumper position on the PCB.

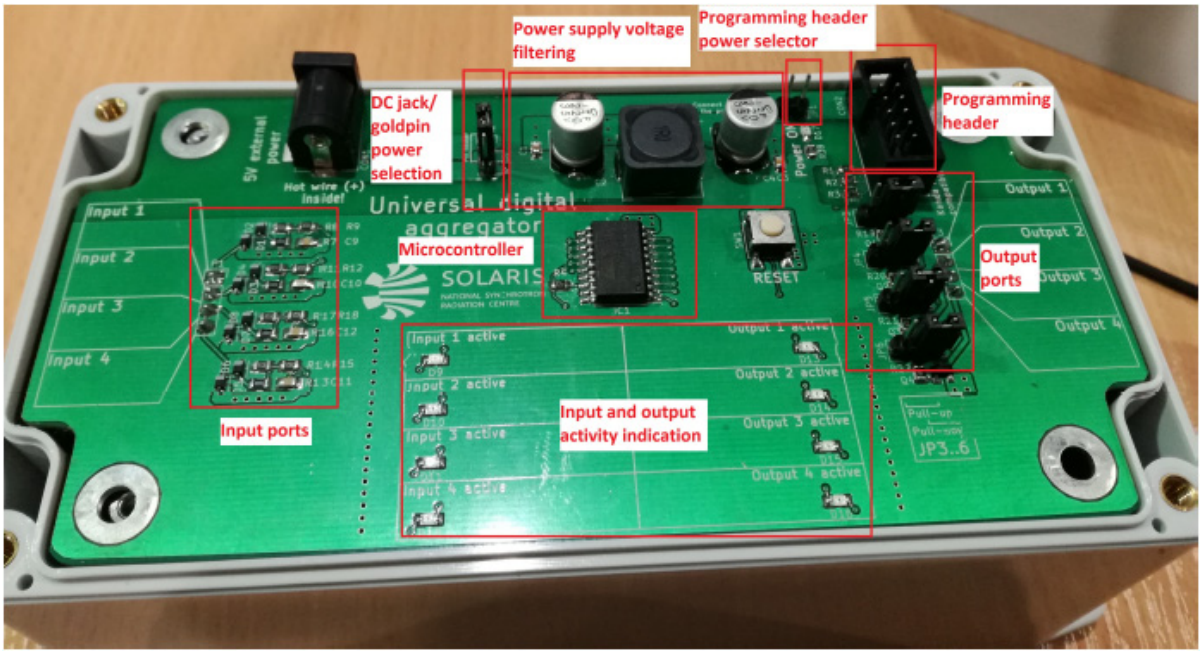

Figure 1: Universal Digital Aggregator PCB.

* maciej.p.kopec@uj.edu.pl

Input Voltage Filtering

Since in-line power is not filtered in any way, an additional LC filter was added to eliminate harmonic noise and to stabilise the input.. The power jack input is also connected to the filter. Schematic of the filter is shown in Fig. 2

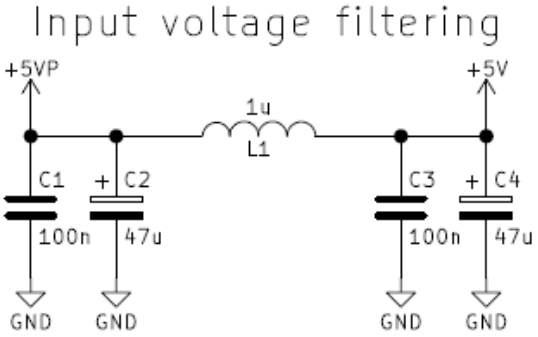

Figure 2: Input power filter schematic.

Input Protection Circuit

For ensuring long and stable operation of the Universal Digital Aggregator it is needed to protect the digital inputs of the microcontroller. The device may be operated using signal transmitted through long cables therefore a good quality of the input signal need to be ensured to protect hardware on the PCB. For that purpose a dedicated circuit for each input is designed, shown in Fig.3. It consists of two Schottky diodes, which clamp any over- or undervoltage. 10Ω resistor is the first stage of filtering the ripples which occur when dealing with signals transmitted over long distances. 10 kΩ resistor acts as a pull-up resistor as the inputs may be connected to switches connected to the ground. 1 kΩ resistor together with 10 nF capacitor are the second stage of ripple filtering and adjusting the input signal. It is simply a low-pass filter with a cutoff frequency of around 1.6 MHz. Together it prevents from over- and undervoltage, overcurrent and voltage ripples which may occur on the inputs.

Output Configuration

In this configuration the transistor acts as a simple switch which, when activated, connects the OUT output to the ground. The jumper (JP3, but also JP4-6) is used to determine which state of the transistor should be default, when the microcontroller is not driving it. This configuration, unfortunately, is not very bulletproof. First of all, it lacks useful circuitry at the transistor's gate. It would be desirable to place a filtering and current limiting resistor in series with the microcontroller output. It would prevent the gate against oxide breakdown even better than the polarizing resistor alone. It would be also good to have some kind of drain current limiting the output. The output transistor is rated to withstand a constant current of 1.3 A, so it is unlikely to break the transistor in such usage.
It is although still possible to damage the output if an unknown device would force a large current through the transistor. This issue could be addressed by installing a small resistor in series, preventing the device from overcurrent.

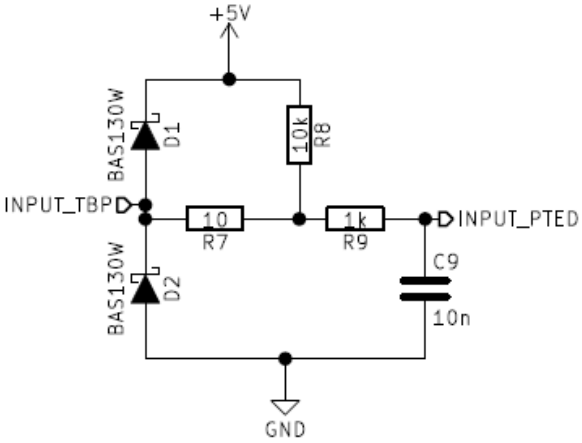

Figure 3: Input protection circuitry designed to clamp any over- and undervoltage and to filter out any distortion coming from long cable transmission and noisy environment.

SOFTWARE

As the Universal Digital Aggregator has a microcontroller as the main executive part, it is fully programmable. It is not possible at the moment to use a custom GUI to load the algorithm to the device. It may be added in future releases. As for now it is necessary to modify and recompile the source code written in C.

Source Code

The software consists of two source files:

- iodef.h,
- main.c.

First contains information about which pins correspond to which "abstract" input (1, 2, 3 or 4) and all the activity definitions to be described later in this section. The second one is the file containing the main loop and the whole logic associated with the input-output relation.

Defining IO states

iodef.h file contains all the necessary information for the program to check and detect activity and react properly by setting active state on the desired output. It contains hardware definitions of the pins corresponding to each abstract input/output and conditions in which the logic states are considered active. When the active state is defined to '1' activity flag will appear in the software whenever the input is logic 1.

Analogically when set to '0' activity flag will appear whenever the input is logic 0. Making the device activity (change of state) sensitive is much more convenient when dealing with groups of signals which have different activity level within, e.g. one limit switch is active low and the other one is active high. The action is needed when both are active (logic states are harder to operate on, rather than activity). Similar behaviour should be programmed on the output. Output should have active logic

state defined in such a way, that the main program sets the activity only, but not the logic state directly.

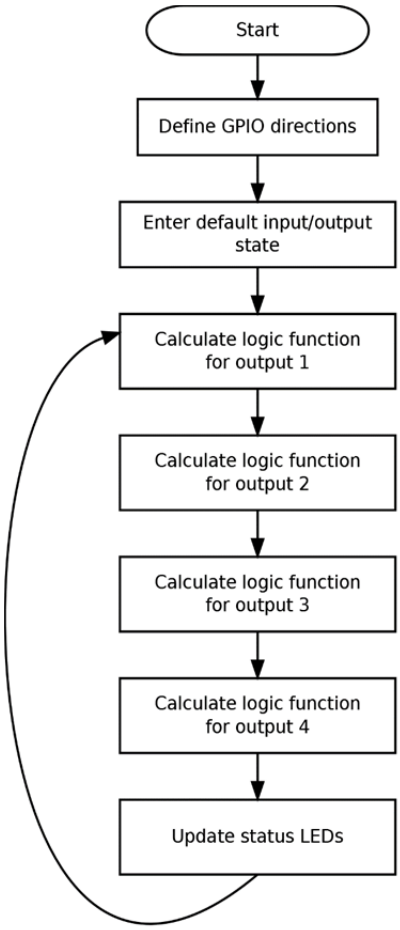

Figure 4: Block diagram of device software processing.

Main Program

The main.c file contains the main loop of the microcontroller and sets everything up. At the beginning of the file there is a declaration of a structure (activity_t) which is filled with bit fields to enable easy access from the language, as well as low memory consumption. This structure is a type of a global activity variable which contains all the latest activities. Then there are instructions for setting up the pins of the microcontroller such as: check the input activity and set the output activity. Afterwards there are instructions which update pins to reflect the activity table.

General Software Block Diagram

Block diagram in Fig. 4 represents the actions taken by the program after booting the device, as well as during normal operation.

CONCLUSIONS

The six-month period of device testing proved reliability and effectiveness that exceeded expectations. Despite being in early prototype stage no faults were observed and it remains operational up to this day in Solaris facility. Further improvement of the device as well as development for other applications will soon follow, such as developing better update algorithm using interrupts and special bitmaps to enable using more user-friendly software which will not require programming skills.