Implementation of a Customisable Readout Sequence for the ALICE ITS Upgrade Explorer Family Chips

Matthias Gazzari

under the supervision of

Felix Reidt

June 28th to September 5th, 2014

1 Introduction

The main purpose of this project is the implementation of a customisable readout sequence to enable partial readouts of the Explorer family chips. These chips are prototypes used in the development of a novel Inner Tracking System (ITS) for the upgrade of the ALICE experiment scheduled for the Long Shutdown 2 (LS2) of the Large Hadron Collider (LHC) in 2018. They are based on Monolithic Active Pixel Sensors (MAPS), which will significantly improve the detector as described in [1, p. 11f]. The prototypes are used to characterise different sensor implementations in order to obtain the best performing sensor for the ITS upgrade.

The layout of the Explorer family chips as shown in Fig. 1 is divided into two matrices: The left one contains 90 times 90 pixels with a pitch of 20 µm and the right one 60 times 60 pixels with a pitch of 30 µm. Each of these matrices are subdivided into nine different regions containing different sensor implementations. Finally, every pixel contains two analogue memories storing the pixel voltage to get rid of common mode noise using Correlated Double Sampling (CDS) as explained in [1, p. 14].

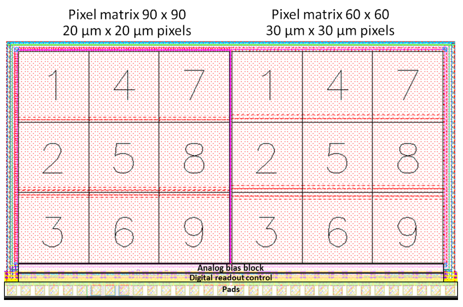

Figure 1: Layout of the Explorer family chip showing nine sectors per matrix, taken from [2].

A typical readout sequence starting with the left- and topmost pixel is shown in Fig. 2 along with the `readout_start_flag` and `readout_end_flag` signals which indicate when it is valid to read out data. Note that these flags are shifted in time compared to the analogue output `outbuf` due to signal propagation delays and the pipeline inside the ADC which samples the analogue output of the Explorer.

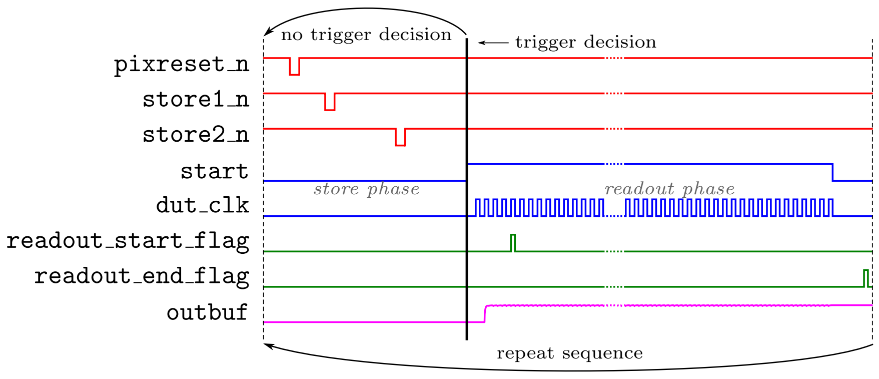

Figure 2: Sketch of a full readout sequence, derived from Fig. 4 in [2].

In order to define the voltage of the collection electrodes inside the pixels, one must set the active low signal `pixreset_n` followed by storing the current potential into the memories using the active low signals `store1_n` and `store2_n` respectively as seen in the *store phase* of Fig. 2. After storing the voltage, one can read out the memories at any time point afterwards. Typically, the *store phase* is repeated until a positive trigger decision is received.

In order to read the previously stored values, the memories are read out sequentially using the two signals `start` and `clock` as seen in the *readout phase* of Fig. 2. Every rising edge of the `clock` signal causes the Explorer chip to advance to the next memory as long as the `start` signal is enabled. Disabling the `start` signal stops and resets the readout sequence to the first memory in the left- and topmost pixel.

The time required to read out the whole chip is determined by the number of memories divided by the readout clock frequency. In this case a clock frequency of 10 MHz leads to a readout time of 2.34 ms using Eq. 1.

$$\frac{(90 \cdot 90 + 60 \cdot 60) \cdot 2}{10\,\mathrm{MHz}} = 2.34\,\mathrm{ms} \tag{1}$$

Because of the long readout duration pixels later in the readout sequence have a higher contribution of shot noise caused by leakage in the analogue memories. To allow a comparable characterisation of every sector one must ensure to start a readout directly at sectors of interest to reduce the noise caused by the readout. Since the Explorer is designed to be read out in columns, the goal is to implement a procedure reading out columns of sectors, e.g. sector 7, 8 and 9.

2 Implementation

2.1 Simulation Model of the Explorer Chip

In order to ease the task of finding a suitable algorithm, a simulation model of the Explorer family chip was created to verify the procedure. However, the simulation model only takes the *readout phase* of the Explorer chip into account and does not cover signals used during the *store phase*. It is only used to show which memory is read out at which location at any given time.

2.2 Defining the Readout Sequence

Subsequently, a parametrised configuration for the Explorer driver was deployed that iteratively creates the requested readout sequence. In order to start the actual readout at the desired column of sectors a third phase called the *forward phase* is introduced as shown in Fig. 3. Its only purpose is to move the memory pointer of the Explorer just in front of the memory to start with. Another difference and possible advantage of the new readout sequence compared to the former one shown in Fig. 2 is that the new one may have a shorter *readout phase* because it only reads out the desired columns of sectors.

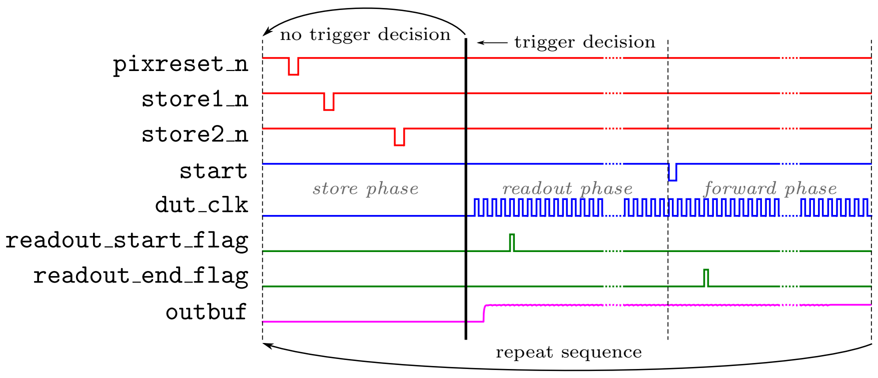

Figure 3: Sketch of a partial readout sequence.

Basically, the whole readout starts with the first *forward phase* followed by the *store* and *readout phase* of the second pass. After the *forward phase* the `start` signal is hold until the following *readout phase* is performed in order to not reset the memory pointer to the first pixel.

Note that in case a readout starts at the first pixel no forwarding is used so that basically the old readout sequence as shown in Fig. 2 is performed. The only difference is that the new readout may stop before reaching the end of the Explorer chip.

2.3 Adaptation of the Explorer Driver

As mentioned in the previous section the new readout actually starts with the first *forward phase* and both preceding phases should have no effect. The first trigger decision as shown in Fig. 3 must be omitted to proceed to the next phase and the readout flags must not be set during the first pass.

The reason for this is that the data taken during the first *readout phase* will most likely be invalid without a preceding *forward phase*. Therefore, the Explorer driver is enhanced in a way that allows to omit the first *store* and *readout phase*.

2.4 Adaptation of the Data Acquisition

Prior to this work the size of the data received was assumed to be constant because every readout had exactly the same data size. Since the algorithm described previously allows for different data sizes it must be ensured that the data acquisition still works. The evaluation whether data is damaged or not and the extraction of the data into a ROOT tree are affected by the adaptations described below.

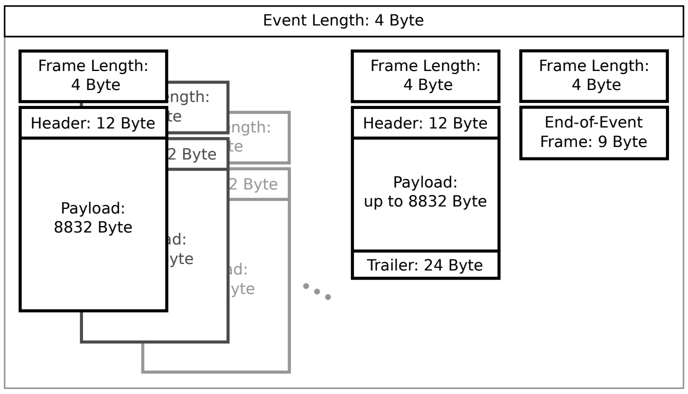

Figure 4: One data set containing several UDP datagrams obtained during a readout.

The data to be processed as shown in Fig. 4 is composed of several UDP datagrams, each preceded by an additional header of 4 bytes and stored within a container, which also has a header of 4 bytes. Every datagram besides the End-of-Event, which is sent afterwards to mark the end of the data, can hold up to 8832 bytes of data.

Note that one readout includes four Explorer chips in parallel with their data already digitised by a 12 bit ADC. Therefore, each memory requires 6 bytes of storage and thus each pixel twice the amount (12 bytes).

In order to evaluate the validity of the data and to extract it correctly, its size is determined empirically. All possible correct data sizes are obtained using the Eqs. 2 to 6. Lower case variables are dimensionless, while capitalised variables are in units of bytes.

$$P = 12 \cdot \left(i\,\frac{90 \cdot 90}{3} + j\,\frac{60 \cdot 60}{3} \right) \quad \text{with} \quad i, j \in \{0,1,2,3\} \wedge i + j > 0 \tag{2}$$

$$n = \left\lfloor \frac{P}{8832} \right\rfloor \tag{3}$$

$$M = 4 + 12 + (P \bmod 8832) \tag{4}$$

$$D = n \cdot (4 + 12 + 8832) + M + 24 + (4 + 9) + 4 \tag{5}$$

$$= n \cdot 8848 + M + 41 \tag{6}$$

Equation 2 is used to iteratively test how many columns of sectors of the first and second matrix are in the received data. Note that the information which columns of the chip are read out is not available in the data. Thus, and for the sake of simplicity the collected data is aligned to the middle of the Explorer chip.

After obtaining a possible linear combination of columns of sectors, one has to determine how the data is split into several datagrams. The number n in Eq. 3 defines how many datagrams are sent with the largest possible payload of 8832 bytes and the size M in Eq. 4 defines the number of bytes in the second to last datagram. The data size D is determined in Eqs. 5 and 6 adding headers, trailers and the End-of-Event datagram to the previously determined datagram sizes.

3 Results

The histograms shown in Figs. 5 and 6 are obtained during short test runs of the new readout sequence in order to verify the implementation described above. These histograms show on odd rows the pedestal data and on even rows the corresponding noise for either the left matrix (20 µm pitch) or the right matrix (30 µm pitch). The first and second columns of these figures show the histograms of the first and second memories respectively and the last column shows the corresponding CDS histograms.

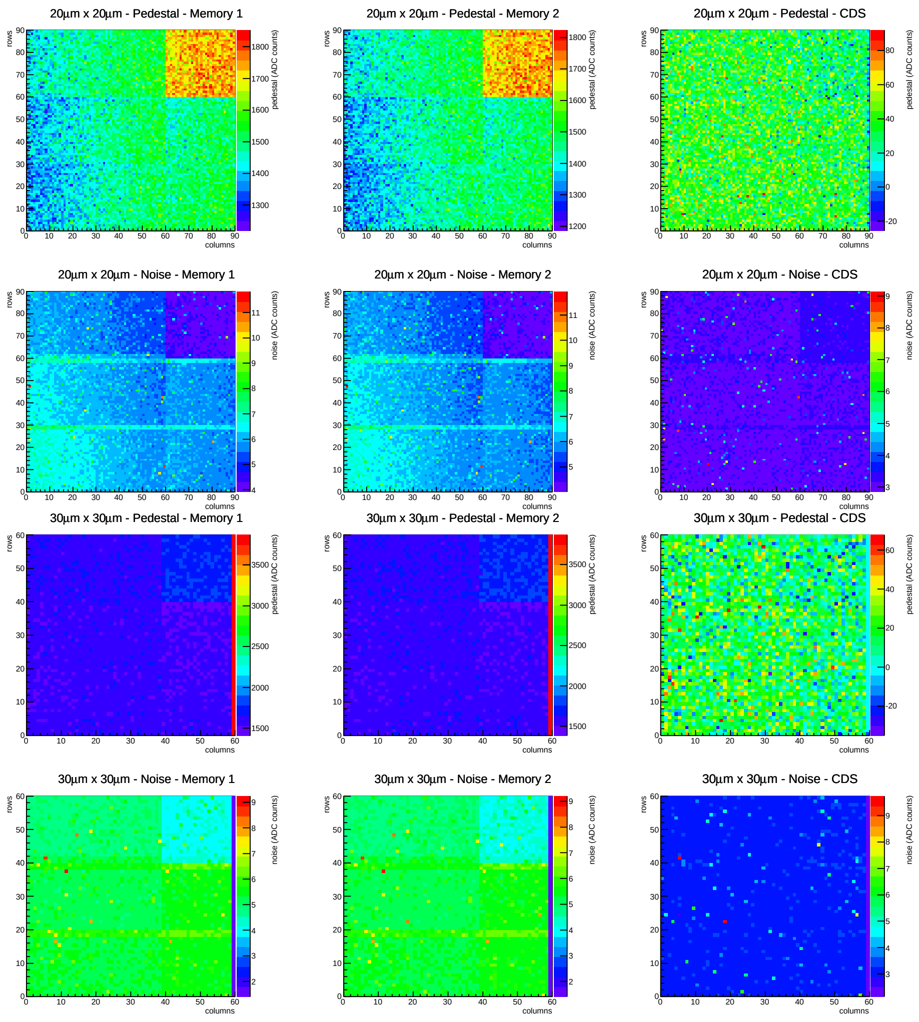

Figure 5: Histograms showing a full readout.

The histograms of a full readout are shown in Fig. 5. As expected, the whole histograms are filled with non-zero data points and CDS improves the noise. Additionally, one can visually identify the different sectors inside of the pedestal and noise histograms in order to verify the correct alignment.

Furthermore, the last column in the second matrix has a high pedestal value compared to other regions on the histograms. This is simply caused by the fact that the last column of the Explorer is broken and serves as an additional visual feedback on the correct readout procedure.

The histograms of a partial readout of the last column of sectors (7, 8 and 9) in the second matrix are shown in Fig. 6. In contrast to the previous results the 20 µm pitch histograms are omitted because, as expected, they are blank. Although the rightmost column of sectors is read out the histograms are plotted at the leftmost position within the matrix because the information

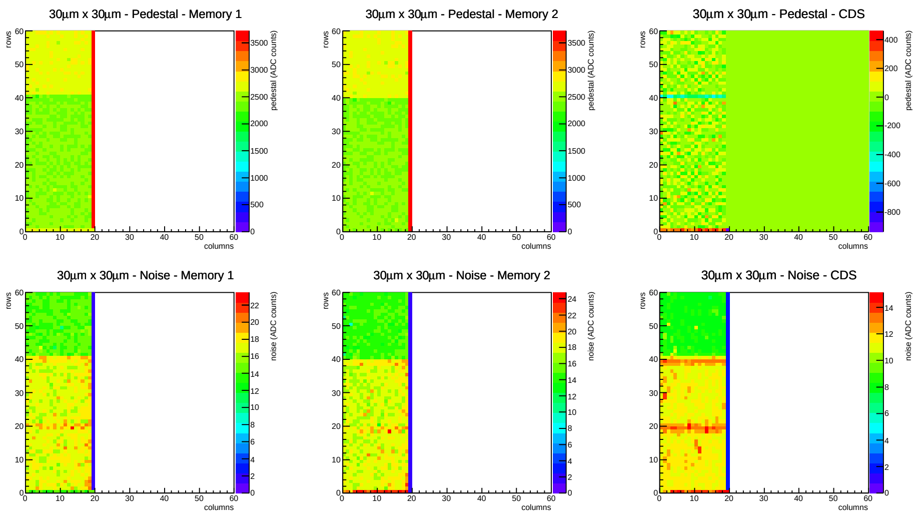

Figure 6: Histograms showing a partial readout of the sectors 7, 8 and 9 of the second matrix.

which columns of the chip are read out is not available in the data (cf. Sec. 2). However, one can visually identify the broken column as shown before in Fig. 5 to confirm that the last column of sectors is actually read out.

Additionally, one can see that the histograms of the first memory are off by one pixel compared to the ones of the second memory or even compared to the full readout seen before.

4 Summary and Outlook

Using a simulation model of the Explorer family enabled a successful implementation of a customisable readout sequence to read a specified portion of the Explorer chip. To make a simple solution possible the Explorer driver had to be adapted to skip the first *store* and *readout phase* along with the first trigger decision. Furthermore, the processing of the collected data had to be adapted to cope with different data sizes obtained during different readouts.

However, a minor flaw regarding the partial readouts as shown in Fig. 6 remained (where the histograms of the first memory are off by one pixel) and should be fixed in the future. Most likely, the *forward phase* is slightly too short so that the actual readout starts at the second memory in front of the first pixel specified to be read out. Consequently, it is assumed that the histograms of the first and second memories are swapped and the actual readout stops just in front of the second memory of the last pixel specified.

Additional evidence for this hypothesis is that the full readout shown in Fig. 5 that has no *forward phase* is not faulty. If the *forward phase* is left out the readout always starts at the correct memory.

The newly developed readout sequence provides means to compare individual sectors with lower shot noise. To take advantage of this method more data has to be collected in order to ease the task of finding the best performing sensor type to be used in the upgrade of the ITS.

References

[1] ALICE Collaboration. Technical Design Report for the Upgrade of the ALICE Inner Tracking System. *J.Phys.*, G41(8):087002, 2014.

[2] C. Cavicchioli et al. User Manual of the ALICE Matrix 1 of the ITS CERN submission. 2012. Internal Document.