PAPER

Observation data pre-processing and scientific data products generation of POLAR

View the article online for updates and enhancements.

*R*esearch in
*A*stronomy and
*A*strophysics

Observation data pre-processing and scientific data products generation of POLAR

Zheng-Heng Li[1,2], Jian-Chao Sun[1], Li-Ming Song[1], Bo-Bing Wu[1], Lu Li[1], Xing Wen[1,2], Hua-Lin Xiao[1,3], Shao-Lin Xiong[1], Lai-Yu Zhang[1], Shuang-Nan Zhang[1] and Yong-Jie Zhang[1]

[1] Key Laboratory of Particle Astrophysics, Institute of High Energy Physics (IHEP), Chinese Academy of Sciences, Beijing 100049, China; *Zhengheng.Li@ihep.ac.cn*
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract POLAR is a compact space-borne detector initially designed to measure the polarization of hard X-rays emitted from Gamma-Ray Bursts in the energy range 50–500 keV. This instrument was launched successfully onboard the Chinese space laboratory *Tiangong-2* (*TG-2*) on 2016 September 15. After being switched on a few days later, tens of gigabytes of raw detection data were produced in-orbit by POLAR and transferred to the ground every day. Before the launch date, a full pipeline and related software were designed and developed for the purpose of quickly pre-processing all the raw data from POLAR, which include both science data and engineering data, then to generate the high level scientific data products that are suitable for later science analysis. This pipeline has been successfully applied for use by the POLAR Science Data Center in the Institute of High Energy Physics (IHEP) after POLAR was launched and switched on. A detailed introduction to the pipeline and some of the core relevant algorithms are presented in this paper.

Key words: gamma-ray burst: general — methods: data analysis — instrumentation: polarimeters

1 INTRODUCTION

POLAR (Produit et al. 2005) is a compact space-borne detector which is specially designed and dedicated to measuring the polarization of hard X-ray/Gamma-Ray emissions from Gamma-Ray Bursts (GRBs) in the energy range 50–500 keV. POLAR uses plastic scintillators (PSs) as the main detection material along with the multi-anode photomultiplier tube (MAPMT) to measure the polarization by recording the distribution of azimuthal scattering angle of the incoming photons interacting with the detection material through the process of Compton scattering (Xiong et al. 2009). The POLAR instrument consists of two major components, the the Outer BOX mounted on the outer surface of *Tiangong-2* (*TG-2*) (OBOX) and the Inner BOX mounted in the interior of *TG-2* (IBOX), the details of which are described in Produit et al. (2018). POLAR was launched onboard the Chinese space laboratory *TG-2* on 2016 September 15 and switched on successfully a few days later. Then, a series of in-orbit tests were performed and the normal science data acquisition was started subsequently, during which time tens of gi-

gabytes of detection raw data were produced in-orbit by POLAR and transferred to the ground every day. The directly downloaded raw data from POLAR consist of three types, which are SCI_0B, AUX_0B and ENG_0B. The raw data of type SCI_0B contain the scientific detection data of POLAR and the engineering data are stored in the raw data of type AUX_0B and ENG_0B. All the raw data from POLAR are in the form of a data stream of binary packets generated by the Field-Programmable Gate Array (FPGA) which are not easy to analyze directly for science purposes. Therefore, pre-processing all the detection raw data and generating well-organized, high level scientific data products as quickly as possible are the first important steps and are required by the POLAR experiment for the following science analysis, such as data quick-look, in-orbit calibration (Li et al. 2018), polarization analysis for the detected GRBs, etc. The requirements of data pre-processing and scientific data products generation for POLAR mainly include:

– iterating and decoding data packets as well as converting engineering data from binary values to physical values;

– doing time alignment between packets from the Center Trigger computer (CT) and the Front-End Electronics (FEE) in modules for each event;
– reconstructing the absolute time of each event;
– doing the coordinate transformation to calculate the direction of POLAR in celestial coordinates (J2000);
– doing proper data merging and splitting to organize all different kinds of data into a simple data structure which is friendly for science analysis.

A full pipeline and the related software to automatically do these works were designed, developed and tested before the launch date of POLAR and fully verified during the in-orbit test phase of POLAR after launch. After the raw data of POLAR arrive at the Institute of High Energy Physics (IHEP), the three types of raw data are firstly decoded separately, and for the scientific detection data the time alignment for physical events are also performed in the decoding phase. Then using the decoded engineering data the absolute time of each event can be reconstructed. Finally, for convenience of scientific analysis, all of the different kinds of data are merged together with a simple data structure, keeping only the necessary data fields that are directly needed by science analysis. The merged data are called SCI_1Q and are used as the input and standard data format for the science data analysis pipeline. Based on the data product of SCI_1Q, some higher scientific data products for publication can also be generated concerning the data archiving and scientific data products of some specific celestial events like GRB and solar flare events.

As a part of the basic software suite of the POLAR Science Data Center (PSDC) at IHEP, this pipeline has been successfully running and providing service for the science data analysis of researchers after POLAR was launched. This pipeline can work automatically without manual intervention and will send an email to notify the researchers that new scientific data products have been generated. Up to now, with scientific data products generated by the pipeline of the current version, the in-orbit instrument performance study and calibration have been finished and are presented in Li et al. (2018), and the first scientific results of POLAR have been produced and published in Zhang et al. (2019) recently. This paper will give a detailed introduction of this pipeline and the main algorithms that are important and specially needed by POLAR. The programming languages C++ and Python are widely used and the ROOT[1] Library developed by CERN is the main tool that was employed in the pipeline to organize the scientific data products. Besides the pipeline, a table describing the main data structure specification for the high level scientific data products of POLAR in ROOT files will also be presented in this paper.

2 RAW DATA FROM POLAR

The scientific detection data and engineering data from POLAR are firstly collected online by and temporarily stored on the *TG-2* platform, then transferred via telemetry to the ground station where all the data will be transferred to the Payload Operation & Application Center (POAC) of *TG-2* according to the data transmission schedule every day. The raw data from POLAR are finally transferred from POAC to IHEP through FTP and synchronized from IHEP to the cooperating institutes in Switzerland by rsync[2]. All the raw data generated by POLAR directly downloaded from *TG-2* are called the 0B level data, all of which are binary data. The 0B level data of POLAR have three types which are defined as SCI_0B, AUX_0B and ENG_0B respectively and they are stored separately in different files. The 0B level data files are right at the start and the initial input of the data pre-processing and scientific data products generation pipeline of POLAR. The three types of 0B level data are firstly summarized in Table 1, then described in detail in the subsections.

2.1 SCI_0B Data

All the scientific detection data of POLAR are stored in raw data files of type SCI_0B. The majority of the scientific detection data is the physical event data collected from the MAPMT. This mainly includes the energy value of each channel with the unit of Analog-to-Digital Converter (ADC) channel, the trigger information of each channel, the time information of each event which is the local time of the instrument, etc. As discussed in Produit et al. (2018), POLAR consists of 25 standalone modules with their own FEE at the bottom side and a CT under the OBOX aluminum frame. For each physical event, there will be a trigger packet generated by the CT and one or more module packets generated by the corresponding triggered modules which have channels with energy deposition higher than the hardware threshold. The trigger packet has the information on which modules are being triggered and therefore are having data packets generated, while the module packets contain the information on which channels are being triggered. As each detected physical event has multiple data packets from both CT and FEEs and those packets are stored separately in the raw data file, there is a requirement to align those data packets that belong to the same event after the packet iterating and decoding. The alignment between the trigger packet from CT and the module packets from FEEs is the basis for the event reconstruction from

Table 1 Summary of Raw Data from POLAR

Type	Packet Contents	Packet Frequency	Processing Requirements
SCI_0B	Detection data of physical and pedestal events: trigger bits, energy deposition, timestamp, etc.	Depending on event rate	Decoding, time alignment between trigger packets and module packets, absolute time reconstruction, etc.
AUX_0B	Operation mode, threshold and high voltage setting of each module, temperature of each module, etc.	1 packet every 2 seconds	Decoding, connecting odd packets and even packets, conversion from digital value to physical value, etc.
ENG_0B	Platform parameters data (PPD) of *TG-2*, part of the housekeeping data, feedback of command injection, digital telemetry data	1 packet per second for PPD	Extraction of PPD, decoding, coordinate calculation and transformation for the pointing direction of POLAR, etc.

the raw data of POLAR. The algorithm that performs this alignment for the scientific detection data of POLAR will be presented in Section 3.2.

2.2 AUX_0B Data

All the engineering data related to POLAR OBOX (also called auxiliary data or housekeeping data), such as the high voltage setting of each MAPMT, the temperature of each module measured by the temperature sensor mounted on the FEE, the OBOX operation mode, etc., are stored in raw data files of type AUX_0B. Those engineering parameters are very important and necessary for the science data analysis, especially for in-orbit calibration. The raw data of type AUX_0B also contain engineering data like the command injection feedbacks, the history of executed commands, the current and voltage measurements of electronics, etc. which are only needed for instrument testing and monitoring but not directly needed by later science data analysis. Unlike type SCI_0B whose data rate is dependent on the physical event rate, the data rate of the engineering data from POLAR OBOX is fixed, which is one new engineering data packet from OBOX every 2 seconds. Besides the high voltage setting and temperature measurement, there is another kind of very important information stored in the engineering data of OBOX, that is the matching between the local time value of CT and the absolute GPS time value provided by the Pulse Per Second (PPS) signal from the GPS receiver, which is updated every minute by the synchronization (SYNC) command from IBOX. This matching between the local time and the absolute GPS time is used as the reference to reconstruct the absolute time for each physical event that is stored in the raw data of type SCI_0B. Section 3.4 will discuss the method to reconstruct the absolute time for each event. While for the engineering data packets, the absolute time does not need to be calculated separately because the absolute GPS time value has already been tagged into each engineering data packet when it is generated.

2.3 ENG_0B Data

The raw data of both type SCI_0B and type AUX_0B are transferred by the Low-Voltage Differential Signaling (LVDS) bus communication while the raw data of type ENG_0B are transferred by the separated 1553B communication channel from POLAR IBOX. The raw data of type ENG_0B are the data stream of fixed length (76 bytes) binary packets. Several different types of data are stored in the raw data of type ENG_0B, which are part of the housekeeping data (like the trigger rate of each module) from POLAR OBOX, the feedback of command injections, the feedback of platform parameters and the digital telemetry data. The four types of data are stored in different packets and can be identified from each other by the special code stored in the packet header. The most important data packet is the one containing the feedback of platform parameters, which is the only necessary data from the raw data of type ENG_0B that are needed by the science data analysis. Therefore, only the data packets containing the platform parameters are extracted and decoded. One new data packet with the updated platform parameters is generated every second. The platform parameters mainly include the position/velocity in the World Geodetic System 1984 (WGS-84) coordinate system and the attitude information (yaw/roll/pitch angle) of *TG-2* platform as well as the corresponding UTC time. These parameters are necessary for calculating the pointing direction of POLAR. The pointing direction of POLAR is necessary for the subsequent calculation of the incident angle of GRBs in POLAR's local coordinate system. The incident angle is needed by both the study of localization of a GRB using the data of POLAR (Suarez-Garcia et al. 2010) and the Monte Carlo simulation (Kole et al. 2017) for polarization analysis. The pointing direction of POLAR is first calculated in the WGS-84 coordinate system then transferred to celestial coordinates (J2000) which are widely used in the field of astrophysics. Section 3.5 will discuss the method of the pointing direction calculation and the associated coordinate transformation for POLAR.

3 RAW DATA PRE-PROCESSING PIPELINE

The pipeline for pre-processing the raw data of POLAR is schematically illustrated by Figure 1. The three types of 0B level raw data are firstly decoded by steps ①, ② and ③ respectively. The decoding process for the three different types of data involves several common program modules in charge of iterating the frames and packets within each file and extracting values from each packet according to the format specification of the raw data. Those common program modules will be discussed in Section 3.1. Then several other steps are involved for different purposes after the decoding steps. In this section, a general description of the pipeline for each step will be given first, then the main algorithms involved in those steps as well as additional details will be discussed in the subsections. Finally, a subsection discussing the file organization and automatization of the pipeline will be provided.

① Decoding raw data of type AUX_0B. In this step all the engineering parameters like temperature and high voltage need to be converted from binary data to meaningful physical values after they are extracted. The decoded data of type AUX_0B in the first stage are called the data product of level AUX_1M.

② Decoding raw data of type SCI_0B. In this step, packets of the same event from FEEs should be correctly aligned to the corresponding packet from CT (Sect. 3.2). After they are decoded all the packets belonging to the same event should be properly organized by a particular mechanism (Sect. 3.3) in the decoded data file. The decoded data of type SCI_0B in the first stage are called the data product of level SCI_1M.

③ Decoding raw data of type ENG_0B. In this step, packets containing the PPD need to first be identified during their iteration then applied to the decoding process. The calculation for the pointing direction of POLAR as well as the coordinate transformation (Sect. 3.5) also need to be performed in this step after the platform parameters are extracted and decoded. The decoded PPD from the raw data of type ENG_0B in the first stage are called the data product of level PPD_1M.

④ As mentioned in Section 2, the time of each event in the data of type SCI_1M decoded from type SCI_0B is the local CT time in the POLAR instrument. The data of type AUX_1M decoded from AUX_0B contain the matching between the local CT time and the absolute GPS time which is updated every minute. Therefore, this step is to use the local CT time in SCI_1M and the absolute GPS time reference in AUX_1M to reconstruct the absolute time of each event (Sect. 3.4). The data product after the absolute time of each event is calculated and what is attached is called the data product of level SCI_1P.

⑤ The time range for the data files of type SCI_0B and AUX_0B is always the same as well as the corresponding AUX_1M and SCI_1M (1P) because they are transferred out of POLAR by the same LVDS bus communication channel. While the time range of the data file of type ENG_0B as well as the corresponding PPD_1M is different from that of SCI_0B and AUX_0B as a result of the different data transmission channel which is the 1553B communication channel. In order to merge the three different types of data in step ⑥ conveniently, this step is to align the data time range of PPD_1M with that of SCI_1P and AUX_1M by doing the data splitting and merging on the data file of type PPD_1M. Then the generated data of type PPD_1N have the same time range as that of SCI_1P and AUX_1M.

⑥ This step merges the three types of data, SCI_1P, AUX_1M and PPD_1N, into a single data file with a simpler data structure (Sect. 3.6), where the organization of event data is highly simplified and each event is attached with the engineering parameters (temperature measurement, high voltage setting, etc.) and the platform parameters (position of *TG-2*, pointing direction of POLAR, etc.). In the merging process only the data fields from the three types that are necessary for the science data analysis are kept. The merged data file is called the data product of level SCI_1Q, which is proposed to be the input of the science data analysis like the in-orbit calibration and the polarization analysis.

3.1 Iterators and Decoder

The raw data of all the three types are actually the sequence of binary frames and packets. Each type of frame and packet has their specific headers at the beginning. For types SCI_0B and AUX_0B, the packets also have the Cyclic Redundancy Check (CRC) code at the end. In the middle are the data corresponding to either scientific detection data or engineering data. The headers are used to identify and separate different frames and packets, while the CRC code is used to check if the frames and packets are transferred correctly. For each type of data a separate iterator module is designed to be in charge of iterating the frames and packets sequence and applying CRC to each frame and packet. The responsibilities of the iterator for each data type are listed below. The decoder is designed as a class with several member functions to extract values from the frames and packets provided by the iterators given the range of bytes or bits.

– For the raw data of type SCI_0B, the data of each physical event consist of one trigger packet from the CT
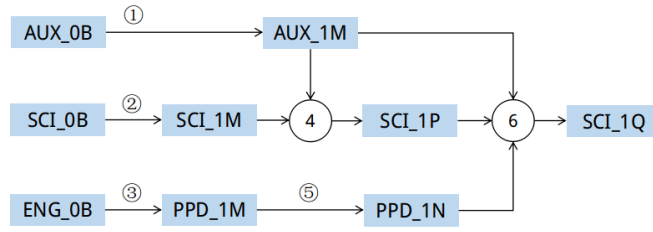
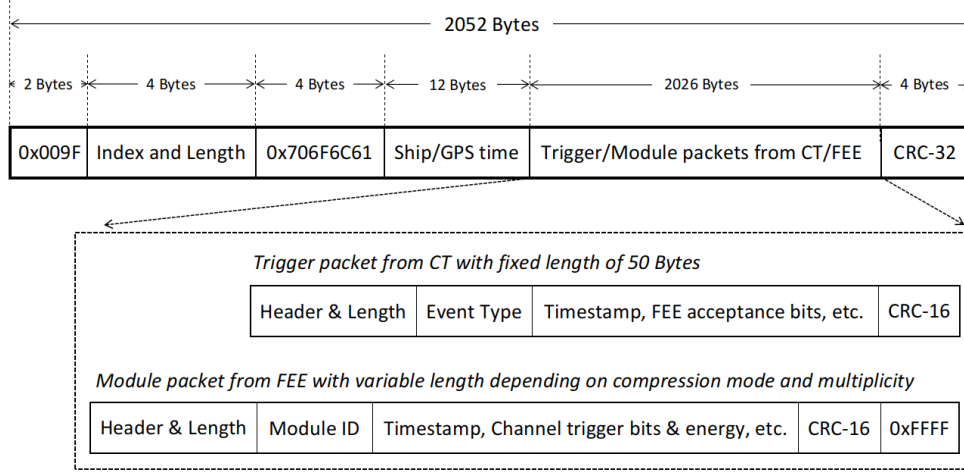**Fig. 1** Raw data pre-processing pipeline for POLAR.

Fig. 2 Structure of the frame and the packed trigger and module packets of SCI_0B data.

and one or more module packets from the triggered FEEs. Those packets are relatively short and can have variable lengths. Those packets are firstly generated in the OBOX then transferred to the IBOX where they are packed into a series of longer frames with a fixed length of 2052 bytes. The structure of the frame and the trigger and module packets packed in it is shown in Figure 2. The iterator for type SCI_0B is therefore responsible for iterating the fixed length frames and the short variable length packets within those frames, connecting the packets that are across two adjacent frames and checking the CRC to tag good or bad packets. For SCI_0B data two different kinds of iterators are designed for frame iterating and packet iterating respectively, and the frame and packet iterating procedure is schematically illustrated by the flow chart shown in Figure 3.

– For the raw data of type AUX_0B, each packet of engineering data from the OBOX is split into two smaller packets in the IBOX which are called the odd packet and the even packet respectively. The length of both is fixed at 260 bytes. These two different packets are transferred out separately from the IBOX every second. The iterator for type AUX_0B is therefore responsible for iterating the fixed length packets, connecting the odd and even packets which belong to the same OBOX packet and doing the CRC checking.

– The raw data of type ENG_0B are simply a sequence of packets with a fixed length of 76 bytes. Therefore, the iterator for type ENG_0B is also very simple, whose responsibility is just to iterate the fixed length packets. During the iteration process, the iterator for type ENG_0B identifies and only keeps the PPD packets for decoding according to the information in the header as only the PPD data are needed for later science data analysis.

3.2 Time Alignment Algorithm for Physical Events

The trigger packets and module packets that belong to different events are mixed together in the raw data. The goal of time alignment is to correctly align the trigger packet and the module packets that belong to the same event. The trigger packets from CT and the module packets from the same FEE are always stored in time order, but this is not guaranteed for the packets from different FEEs and the packets from CT and FEE for different events. The packets of one event have the chance to be transferred out earlier than that of the event before the current event when the physical event rate is high because of the time delay of data transmission between CT and FEE. Therefore, it is hard to
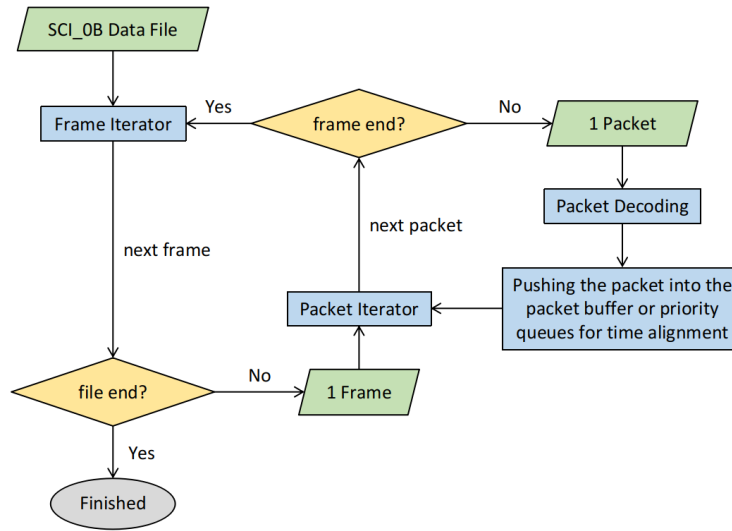
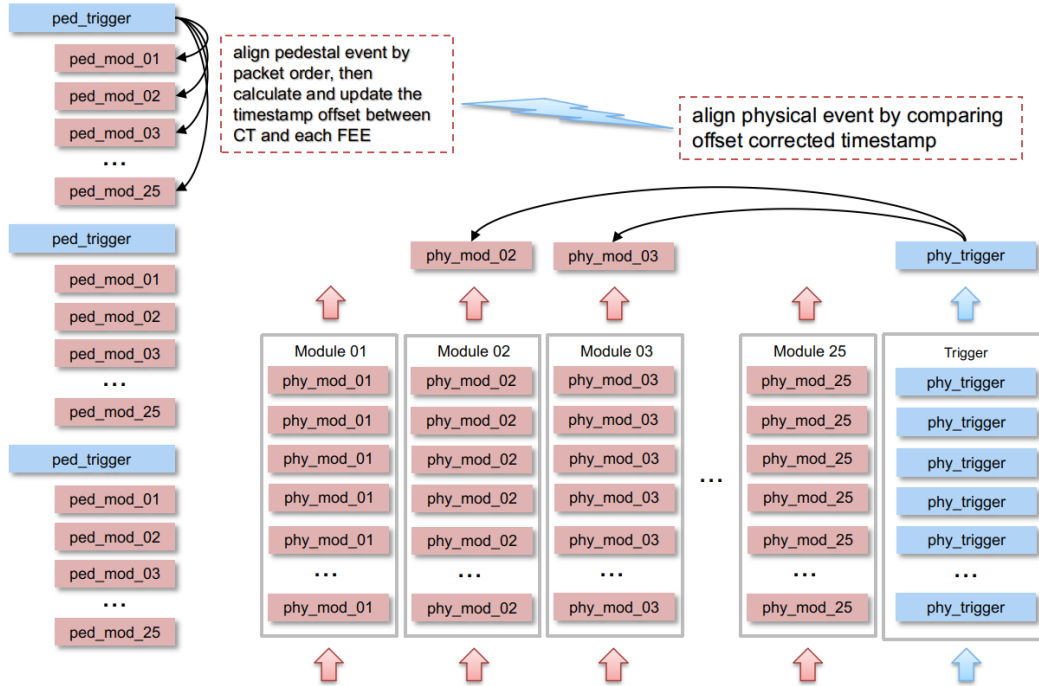**Fig. 3** Flow chart schematically showing the procedure of frame and packet iteration for SCI_0B data.

Fig. 4 Schematic representation of the time alignment algorithm for both pedestal events and physical events, where the *blue boxes* represent trigger packets from CT, *red boxes* signify module packets from FEE and *gray rectangles* mean priority queues. The leftmost column shows the trigger and module packets of pedestal events by the order of packets in the raw data file, and the right columns display the trigger and module packets of physical events after they are pushed into the priority queues.

only use the information of packet order in the raw data file to align the trigger packet and module packets for physical events.

Besides the detected physical event data, the raw data of type SCI_0B also contain the pedestal event data for the purpose of measuring the pedestal level of each channel in the calibration process (Li et al. 2018). Normally, each pedestal event has one trigger packet from CT and one module packet from each FEE, which is a total of 26 packets. As the rate of pedestal events is very low (1 Hz) and the 1 second waiting time between two adjacent pedestal events is far longer than the data transmission time delay, the packets that belong to the same pedestal event are always adjacent to each other in the raw data. It is also found from the raw data that the trigger packet is always transferred out earlier than the module packets for the same

event. Therefore, unlike the case of physical events, for pedestal events it is very easy to align the trigger packets and module packets that belong to the same pedestal event only according to the order of the packets in the raw data as shown by the leftmost column in Figure 4. After the packets of pedestal events are properly aligned, the aligned pedestal events can provide one kind of important information, that is the timestamp offset between CT and FEE. The timestamp offset is caused by the different starting times between CT and FEE. This timestamp offset is needed by the alignment of trigger packets and module packets for physical events based on comparison of timestamps.

For aligning the packets of physical events, two kinds of information can be used. One is the timestamp of the two kinds of packets as mentioned above and the other is the trigger information about the module that is stored in the trigger packet of each event. The timestamp value after the correction of the offset between CT and FEE can be used to check if two packets belong to the same physical event. The trigger information of the module that is stored in the associated trigger packet is one bit for each module to identify if the module is triggered and therefore has a module packet or not in this event. With the two kinds of information, the alignment algorithm for physical events is implemented as below:

– Step 1. Establish 25 priority queues[3] for the module packets from the 25 FEEs and 1 priority queue for the trigger packet from the CT. In each priority queue, the priority level is set using the timestamp, then all the packets in each priority queue are sorted by time where the packet with the earliest timestamp will always be at the top of the priority queue.
– Step 2. During the time that the packets are iterating and being decoded, the decoded packets are pushed into the corresponding priority queues according to the type and module ID of the packet.
– Step 3. As depicted in Figure 4, take the trigger packet at the top of the priority queue of trigger packets, then check the trigger information of the module that is stored in the trigger packet and accordingly select the module packets at the top of the priority queues of module packets for only the modules which are triggered in this event. Lastly, compare the timestamps of those module packets after the offset correction with that of the trigger packet to check if those module packets have timestamps that are similar to that of the trigger packet and therefore belong to this event.

In this method, the situation of packet loss can also be properly coped with according to the difference in the

timestamps between trigger packets and module packets. For example, in Step 3 if the timestamp of one module packet is much later than that of the trigger packet, this can be caused by the loss of the packets by the module, and the module packet with the later (or larger) timestamp should be matched to one of the events after the current event. Therefore, in each trigger packet one extra counter is attached to record the number of module packets that are lost in this event. It should be noted here that in Step 3 the module priority queues should accumulate a certain number of packets before doing the timestamp comparison to avoid an incorrect judgement of packet loss due to the delay of packet transmission between CT and FEEs.

This event alignment method is based on the trigger information of the module that is stored in trigger packets and the timestamp comparison between trigger packets and module packets, therefore this method is also called the time alignment method. According to the processing result using in-orbit data, this method can correctly align more than 99.99% of all the module packets. This is calculated by computing the ratio of the number of module packets which successfully found their corresponding trigger packet by using the priority queue method and comparing timestamps over the total number of module packets for physical events.

3.3 Organization of Event Data

As mentioned above, the relationship between the trigger packet and module packet for one event is a one-to-many relationship. However, the data structures of the two kinds of packets are different and therefore it is hard to store them in the same TTree, which is the main structure to organize data in ROOT. The decoded data in the first stage should contain all the information in the raw data and in order to save data space and to efficiently read the decoded data event by event a special structure is designed to organize the event data in the ROOT file.

As shown in Table 2, two different TTrees are used to store the trigger packets and module packets, the names of which are "t_trigger" and "t_modules" respectively. In each TTree there are some special numbers to maintain the one-to-many relationship between the trigger packet and module packets that belong to the same event. Every valid trigger packet has been given a sequence number which is called "trigg_num". In t_modules the module packets that belong to the same event are stored adjacently and each module packet records the trigg_num of the trigger packet for the event. As TTree supports randomly accessing an entry by the entry number and in order to efficiently read the data event by event, each trigger packet in t_trigger records the position of the corre-

---

[3] *https://en.wikipedia.org/wiki/Priority_queue*

Table 2 Example of the Special Numbers for Organizing Event Data Using Two TTrees

t_trigger				t_modules			
Entry	trigg_num	pkt_start	pkt_count	Entry	trigg_num	ct_num	event_num
0	0	0	2	0	0	2	0
				1	0	3	0
1	1	2	3	2	1	8	0
				3	1	7	0
				4	1	2	1
2	2	5	1	5	2	3	1
3	3	6	2	6	3	6	0
				7	3	7	1
4	4	8	2	8	4	3	2
				9	4	4	0
5	5	10	3	10	5	12	0
				11	5	7	2
				12	5	8	1
6	6	13	1	13	6	3	3
7	7	14	2	14	7	2	2
				15	7	3	4

sponding module packets in t_modules using "pkt_start" and "pkt_count", where pkt_start records the starting entry number of the sequence of module packets stored in t_modules and pkt_count records the number of module packets for the event. The "ct_num" in the module packet is the ID of the module to which the packet belongs, and the "event_num" is a sequence number like the trigg_num but the scope of event_num is within one single module. Table 2 lists some examples of these special numbers for organizing event data using two different TTrees, where each non-empty row means an entry in the TTree.

It should be noted here that this structure of event data organization with two different TTrees is only used for the decoded data after the first stage. This structure is to maintain the relationship between trigger packets and module packets, and at the same time to reserve all the information in the raw data using a scheme that can save data space as much as possible. Afterwards, a much simpler structure to organize the event data will be introduced for the convenience of science data analysis, which will be discussed in Section 3.6.

3.4 Absolute Time Reconstruction

The engineering data of type AUX_0B contain the matching between the GPS time from PPS and the local timestamp of CT. This matching is updated every minute. In order to reconstruct the absolute time of each event, an iterator to read a pair of GPS time and CT timestamp from the decoded AUX data (AUX_1M) one by one is designed as the first step. Then when calculating the absolute time of one event, the GPS-timestamp pair, that is nearest in time to the frame in which the trigger packet of the event is located, can be used. The header of each frame in the raw data of SCI_0B has the GPS time when the frame is generated as shown in Figure 2, therefore it is very easy to calculate the time distance between the frame and the GPS-timestamp pair. Using the GPS-timestamp pair that is nearest in time to the event is mainly concerning the effect of the temperature change on the frequency of the clock in CT. The temperature of the full instrument as well as that of the CT changes periodically due to the position change of the instrument in different orbits. As shown in Figure 5, the frequency of the clock in CT also changes periodically with the change of the temperature of CT. It can also be seen that the frequency of the clock in CT has a clear inverse correlation with the temperature of CT. As the timestamp of CT is given by the clock in CT, in the plot the frequency of the clock in CT is calculated using the two adjacent GPS-timestamp pairs by equation $f = (t_2 - t_1)/(T_2 - T_1)$, where f is the clock frequency, t_2 and t_1 are the two timestamps and T_2 and T_1 are the two GPS times in the two adjacent GPS-timestamp pairs. During the process of absolute time reconstruction, the correction for the effect of temperature on the clock frequency is therefore performed for all the events within every minute that are between two adjacent GPS-timestamp pairs. The correction uses the calculated clock frequency every minute instead of the standard frequency of the clock in CT (12.5 MHz) when computing the absolute time for each event. With the calculated clock frequency f, the absolute time of each event is computed by equation $t_{abs} = (t_{loc} - t_1)/f + T_1$, where t_{loc} is the local timestamp of the event and t_{abs} is the reconstructed absolute GPS time of the event in the unit of second.

Using the absolute time of each event reconstructed with the GPS-timestamp pairs and taking the temperature
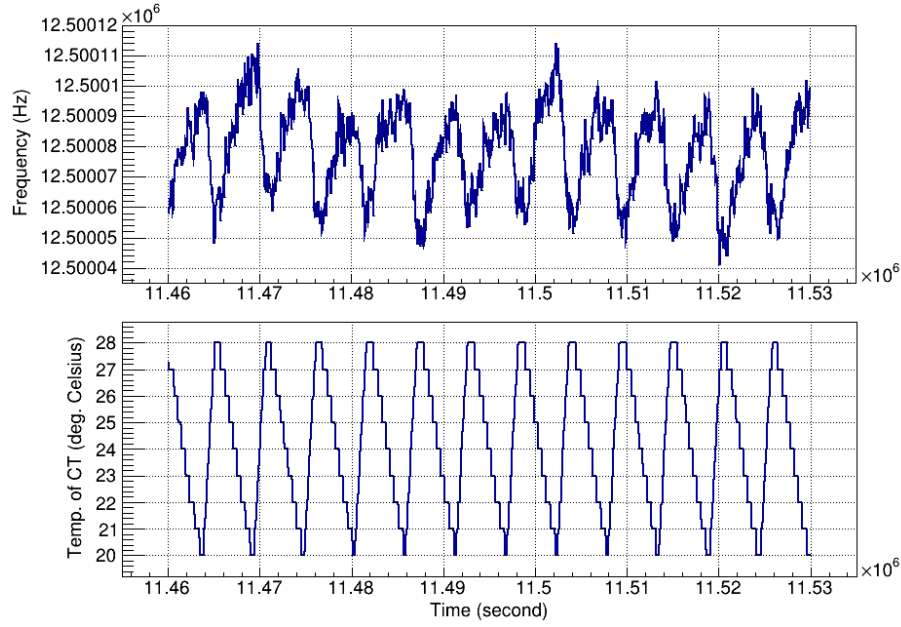
Fig. 5 The curves representing the frequency of the clock in CT (*upper*) and the temperature of CT (*lower*) vs. time.

effect into account, the study of evolution of the spin frequency for the Crab pulsar as presented in Zheng et al. (2017) shows that the spin frequency evolution result for the Crab pulsar achieved using POLAR's data is consistent with that using *Fermi* data, which means that the time system of POLAR is accurate and stable. Zheng et al. (2017) also demonstrates that the sigma of the time residuals for the Crab pulsar observed by POLAR is 84μs, which includes both the error in the reconstructed absolute time for POLAR and also the non-zero red noise of the spin frequency for the Crab pulsar. This means the error of the reconstructed absolute time for POLAR is less than 84μs.

3.5 Pointing Direction Calculation and Coordinate Transformation

The Z and X axes of POLAR are two vectors that are initially defined in the coordinates of *TG-2* after POLAR was mounted. The direction of the Z axis is along while the X axis is perpendicular to the PS bar direction. The directions of the Z and X axes for the POLAR detector need to be calculated in J2000 coordinates. Using the directions of the Z and X axes of POLAR together with the direction of one GRB in J2000 coordinates, the incident angle of the GRB in the detector coordinates can be calculated. The platform parameters that are needed for calculating the directions of the Z and X axes for POLAR in the J2000 coordinate system consist of:

- the attitude parameters of *TG-2*, that are the three Euler angles of yaw, roll and pitch of *TG 2*, which are referred to as Ψ, ϕ and θ respectively,

- the position and instantaneous velocity of *TG-2* in the WGS-84 coordinate system, which are respectively referred to as $\boldsymbol{P} = (x_w, y_w, z_w)$ and $\boldsymbol{V} = (\dot{x}_w, \dot{y}_w, \dot{z}_w)$.
- the Coordinated Universal Time (UTC) corresponding to the recorded platform parameters, which is referred to as t,
- the directions of Z and X axes in the Guidance, Navigation and Control (GNC) coordinate system of *TG2*, which are referred to as $\boldsymbol{k}_g = (x_k, y_k, z_k)$ and $\boldsymbol{i}_g = (x_i, y_i, z_i)$ respectively.

According to the coordinate definitions, the direction of the Z (or X) axis is firstly transformed from the GNC coordinate system ($\boldsymbol{k}_g$) to the orbit coordinate system ($\boldsymbol{k}_o$) by Eq. (1).

$$\boldsymbol{k}_o = \boldsymbol{R}_{go}(\Psi, \phi, \theta) \cdot \boldsymbol{k}_g , \qquad (1)$$

where $\boldsymbol{R}_{go}(\Psi, \phi, \theta)$ is the rotation matrix with the rotating order $Z(\Psi) \rightarrow X(\phi) \rightarrow Y(\theta)^4$. Then the direction of the Z (or X) axis is transferred from the orbit coordinates ($\boldsymbol{k}_o$) to the WGS-84 coordinates ($\boldsymbol{k}_w$) by Eq. (2).

$$\boldsymbol{k}_w = \boldsymbol{R}_{ow}(\boldsymbol{P}, \boldsymbol{V}) \cdot \boldsymbol{k}_o , \qquad (2)$$

where $\boldsymbol{R}_{ow}(\boldsymbol{P}, \boldsymbol{V})$ is the coordinate transformation matrix between the orbit coordinates and the WGS-84 coordinates, which can be calculated using the parameters $\boldsymbol{P}$ and $\boldsymbol{V}$ according to the definition of the orbit coordinates. Finally, together with the UTC, the direction of the Z (or X) axis in the WGS-84 coordinates ($\boldsymbol{k}_w$) needs to be transformed to that in the J2000 coordinates ($\boldsymbol{k}_j$). For

---

4 *https://en.wikipedia.org/wiki/Euler_angles\
#Rotation_matrix*

this transformation, the Python module PyEphem[5] or the C++ library libnova[6] can be used and two basic steps are involved:

- Step 1. calculate the longitude and latitude corresponding to k_w and the sidereal time at the current position (k_w) and UTC (t),
- Step 2. convert the position of k_w in the celestial coordinates of the current epoch to that of epoch J2000.

With the calculated directions of the Z and X axes of POLAR in the J2000 coordinates and the locations of GRBs provided by other instruments like *Swift* and *Fermi*, the computed incident angles of GRBs are as expected by examining the relative count rates of the 25 modules of POLAR for the detected GRBs. The study of the localization of GRBs using POLAR's data is currently ongoing and the localization results of POLAR are consistent with those provided by other instruments within statistical error. This indicates that the directions of the Z and X axes for POLAR in the J2000 coordinates calculated by the procedure presented in this section are correct.

3.6 Data Merging and Reorganization

Before the data merging process will be discussed in this section, the event data, the engineering data and the PPD are stored separately in three different files. The structure for organizing the event data is relatively complicated as discussed in Section 3.3. This multi-file scheme is not convenient for science data analysis. Therefore, a higher level of data product called SCI_1Q is generated based on the three kinds of data which are SCI_1P, AUX_1M and PPD_1N. To generate the data product of level SCI_1Q, the following operations are performed:

- The trigger packet and module packets of each event are merged into a single TTree, and the pedestal events and physical events are also merged together into the same TTree which were previously stored separately in data level SCI_1P.
- The time of each event is converted from GPS time to Mission Elapsed Time (MET) starting from the launch time of *TG-2*. MET is a double-precision floating point value with the unit of second.
- The engineering data and the platform parameter data that are needed for science analysis are attached to each event according to time. The values of the engineering data and PPD are linearly interpolated in time for each event.
- In the merging process only the data fields that are directly needed by science analysis are kept, and the data

fields that are not necessary for science analysis are discarded.
- One sub-level number is attached to each data file of level SCI_1Q to indicate the number of stages that the data file has undergone in the science data analysis pipeline.

The data of level SCI_1Q are designed for the standard science data analysis pipeline as discussed in Li et al. (2018). Even though the data size is approximately doubled comparing with SCI_1P after the merging, with the relatively simple data structure and the single TTree scheme of data level SCI_1Q, the data analysis pipeline software is highly simplified. Some important data fields in the data of level SCI_1Q are listed in Table 3.

3.7 File Organization, Processing Automatization and Email Notification

When in operation, every day about 40 gigabytes, on average, of 0B level raw data are produced in-orbit by POLAR and transferred to the PSDC in IHEP. The raw data files are firstly organized by a strategy using one day as a unit. The raw data transferred to IHEP on the same day will be collected and permanently stored in a single folder for the day and the folder name is the date of the day. The raw data transferred in the next day will be stored in a new folder. This strategy can safely keep all the old data files intact on the PSDC server when continuously receiving new data files from POAC day by day.

For the high level data products generated by the pipeline shown in Figure 1, the file organization uses a different strategy. All data files with the same type, such as SCI_1M, AUX_1M, PPD_1M, SCI_1P, PPD_1N and SCI_1Q, are stored in the same folder with the type name, no matter which day the files are generated. The name of each data file has the exact time range of the internal data with a precision of 1 second. This strategy is chosen for the convenience of processing in the pipeline's automatization as well as for users easily finding the data files corresponding to a specific type within a given time range.

Manually finding and processing all the new raw data files everyday are very time consuming. Therefore, a series of scripts written by Python is prepared to automatize the whole data processing pipeline from 0B level to 1Q level as presented by Figure 1. Those scripts automatically start to run at some specific time points everyday, coordinating with the schedule of data transferring from POAC to IHEP. Once those scripts start to run they will firstly identify the new raw data files in the raw data folder from the current day by comparing file names. Subsequently, they will invoke the data pre-processing programs written by C++ to generate the high level data products for the new

---

[5] *http://rhodesmill.org/pyephem*
[6] *http://libnova.sourceforge.net*

Table 3 Some Important Data Fields in the Data of Level SCI_1Q

Type*	Name	Description
Long64_t	event_id	local sequence number of the event
Double_t	event_time	MET time of the event
Int_t	type	in-orbit recognized type of the event
Bool_t	is_ped	indicates if the event is a pedestal event
Bool_t[25]	trig_accepted	triggering status of each module
Bool_t[25]	time_aligned	time alignment status of each module
Int_t	pkt_count	number of module packets for the event
Int_t	lost_count	number of lost mod. packets for the event
Bool_t[25][64]	trigger_bit	triggering status of each channel
Int_t	trigger_n	total number of triggered channels
Int_t[25]	multiplicity	number of triggered channels for each module
Float_t[25][64]	energy_value	energy deposition value of each channel
UShort_t[25][64]	channel_status	energy readout status of each channel
Float_t[25]	common_noise	common noise of each module
Int_t[25]	compress	data compress mode of each module
Float_t[25]	fe_hv	HV setting of each FEE (from AUX)
Float_t[25]	fe_temp	temperature of each FEE (from AUX)
Double_t[3]	wgs84_xyz	position of *TG-2* (from PPD)
Double_t[2]	det_z_radec	direction of POLAR Z axis (from PPD)
Double_t[2]	det_x_radec	direction of POLAR X axis (from PPD)

`https://root.cern.ch/root/html528/ListOfTypes.html.`

raw data and put them into the corresponding folders. For each file of the high level data products, the command to generate it and the screen output as well as the log output are all reserved in specific folders for later checking or regenerating the file when needed. After all the new raw data files are finished, to generate the corresponding high level data products by the automatization scripts, one notification email will be generated based on what has been processed and sent by the built-in SMTP Python client to researchers in the POLAR group who can access the data products on the PSDC server in IHEP. The notification email mainly includes the starting and finishing time for processing the newly found raw data, a list of all the new raw data files that have been processed, a list of all the new corresponding high level data product files that have been generated and information on any errors that occurred during the processing. According to the notification email, the researchers can login the PSDC server and analyze the interesting data files on the server and the staff in charge of maintaining the data products deal with the exceptions and errors found during the processing. With the automatic scripts, the full pipeline from the 0B level raw data to the 1Q level data product as presented by Figure 1 can be finished within 3 hours for 10 gigabytes of raw data files after they arrive at IHEP. This is efficient enough for the following science data analysis.

This raw observation data pre-processing and high level scientific data products generation pipeline and the automatic scripts introduced in this section have already continuously worked for several months without many manual interventions on the PSDC server in IHEP for POLAR. The core software units used in this data processing pipeline, such as the decoding and time alignment program for SCI_0B data, the decoding program for AUX_0B data, the absolute time reconstruction program, etc., were fully tested and extensively used by processing the ground calibration data and firmware testing data before the launch of POLAR. The full pipeline was validated well by the success of automatically processing all the in-orbit data from POLAR and providing service for the science data analysis of researchers after POLAR was launched.

4 HIGHER LEVEL SCIENTIFIC DATA PRODUCTS

With the data of level SCI_1Q, most of the science analysis for POLAR can be performed. However, concerning the future data publication as well as the convenience of the internal use of the data from POLAR, some higher level scientific data products can be designed and generated. Generating the higher level scientific data products mainly consists of two aspects: data archiving and the scientific data products of specific celestial sources. For data archiving, it needs to recombine the data files of level SCI_1Q in order to remove the data duplication among different files and to organize each data file with a fixed short time length, for example, making each data file have one hour of data. For the scientific data products associated with specific celestial sources, it needs to firstly extract the data segment of SCI_1Q at the time when one event, like a GRB

or solar flare, occurred and was detected by POLAR, then add some property information about the event, such as the T0, T90, location in the J2000 coordinates, incident angle and the energy spectrum parameters of the source. The energy spectrum parameters can be either given by other instruments or provided by the analysis using the data from POLAR itself. As the FITS[7] file format is widely used in the field of astronomy, either the FITS files or the tools to convert the ROOT files to FITS files for each type of higher level scientific data products can also be provided when they are published. The process of generating the higher level scientific data products will not involve some complicated procedures besides the simple procedures like data splitting and merging for the data files of SCI_1Q and the addition of some extra meta-information related to the source, therefore, this section will not give more details about this process.

5 SUMMARY AND CONCLUSIONS

POLAR is a space-borne GRB polarimeter onboard the Chinese space laboratory *TG-2* which was successfully launched on 2016 September 15. In order to pre-process the in-orbit data from POLAR that are transferred from *TG-2* to the ground as quickly as possible and generate the high level scientific data products that are suitable for later science analysis, a full software pipeline was designed and developed before the launch date of POLAR and this paper presented a detailed introduction to the pipeline. The three types of raw data from POLAR and the pre-processing requirements are firstly introduced. Then each step of the pipeline is discussed as well as the relevant algorithms and methods that are used in those steps. The most important data are the event data detected by the POLAR detector, and the event data are converted by four steps: SCI_0B → SCI_1M → SCI_1P → SCI_1Q. The scientific data product of level SCI_1Q simplified the data structure of the event data and merged the necessary engineering data and PPD that are respectively decoded from the raw data of types AUX_0B and ENG_0B. The SCI_1Q data product is chosen as the input and as a standard data format in the science data analysis pipeline in the PSDC at IHEP. Some higher level scientific data products for the data archiving and specific events like GRBs and solar flare events can be generated based on the data product of level SCI_1Q. The in-orbit observation data pre-processing and scientific data products generation pipeline and the related software can

work automatically without manual intervention for most of the time after the raw data from POLAR arrive at IHEP and have been continuously and successfully used for months by the PSDC in IHEP after POLAR was launched and switched on. In practice, the full pipeline from the 0B level raw data to the 1Q level data product for the science data of POLAR can be finished within 3 hours for 10 gigabytes of raw data files after they arrive at IHEP, which is efficient enough for the following science data analysis. Also, after the pipeline for all the newly-arrived raw data has finished, a notification email describing the generation of new scientific data products will be automatically sent to the researchers who can access the scientific data products. With the scientific data products generated by this pipeline, the in-orbit calibration of POLAR has been finished and published so far, and the first scientific results of POLAR have been produced and published.

References

Kole, M., Li, Z. H., Produit, N., et al. 2017, Nuclear Instruments and Methods in Physics Research A, 872, 28

Li, Z., Kole, M., Sun, J., et al. 2018, Nuclear Instruments and Methods in Physics Research A, 900, 8

Produit, N., Barao, F., Deluit, S., et al. 2005, Nuclear Instruments and Methods in Physics Research A, 550, 616

Produit, N., Bao, T. W., Batsch, T., et al. 2018, Nuclear Instruments and Methods in Physics Research A, 877, 259

Suarez-Garcia, E., Haas, D., Hajdas, W., et al. 2010, Nuclear Instruments and Methods in Physics Research A, 624, 624

Xiong, S., Produit, N., & Wu, B. 2009, Nuclear Instruments and Methods in Physics Research A, 606, 552

Zhang, S. N., Kole, M., Bao, T. W., et al. 2019, Nature Astronomy, 3, L258

Zheng, S., Ge, M., Han, D., et al. 2017, Scientia Sinica Physica, Mechanica & Astronomica, 47, 099505

---

[7] *https://fits.gsfc.nasa.gov*