User's Manual
Digital X-ray Processor

Model DXP-2X
Revision A

X-ray Instrumentation Associates

8450 Central Ave.
Newark, CA 94560 USA

Tel: (510) 494-9020; Fax: (510) 494-9040
http://www.xia.com/

User's Manual

Digital X-ray Processor, Model DXP-2X, Revision A.

X-ray Instrumentation Associates
8450 Central Ave.
Newark, CA 94560 USA
Tel: (510) 494-9020; Fax: (510) 494-9040
http://www.xia.com/

1. Overview:

XIA's Digital X-ray Processor (DXP) is a high rate, digitally-based, multi-channel analysis spectrometer that is particularly well suited for EXAFS and other energy dispersive x-ray measurements using multi-element detector arrays. The DXP offers complete computer control over all amplifier and

spectrometer controls including gains, peaking times, and pileup inspection criteria. The DXP's digital filter typically increases throughput by a factor of two or more over available analog systems at comparable energy resolution but at a lower cost per channel. The DXP's full computer interface allows all data taking and calibration operations to be automated for multi-element detectors, thus greatly reducing the possibility of human error. The DXP is easily configured to operate with a wide range of common detector/preamplifier systems, including pulsed optical reset, transistor reset, and resistive feedback preamplifiers. The DXP –2X Model C combines four channels in a single width CAMAC module. The DXP-2X Model 4T is an enhanced version with an external timing input for special purpose experiments including both time resolved and phase-locked spectroscopy.

1.1. DXP-2X Features:

- Single CAMAC module replaces 4 channels of spectroscopy amplifier and pulse processing electronics at significantly reduced cost.
- Operates with a wide variety of x-ray detectors using preamplifiers of pulsed optical reset, transistor reset or resistor feedback types.
- Maximum throughput over 500,000 counts/sec per channel.
- Programmable peaking times between 0.125 and 40 μsec.
- Covers energy range from light elements (B K-α, 185 eV, with appropriate detector) to γ-rays (10 MeV or more).
- Pileup inspection criteria computer selectable, including fast channel peaking time, threshold, and rejection criterion.
- Accurate ICR and livetime reporting for precise deadtime corrections.
- Multi-channel analysis for each channel, allowing for optimal use of data to separate fluorescence signal from backgrounds.
- Enables automated gain setting and calibration to facilitate tuning multi-element detector systems.
- External Gate allows data acquisition on all channels to be synchronized.
- External Sync (Model T only) allows time resolved data to be collected.

1.2. Major differences between the DXP-2X and the DXP-4C:

The DXP-2X is XIA's second generation Digital X-ray Processor. It has been designed to be a significant performance upgrade while functionally compatible with its predecessor, the DXP-4C, which has been produced since 1996. Though the two products are not firmware compatible (since they use different DSP processors), the host software is nearly identical.

The main differences between the two products are:

- The DXP-2X uses a newer higher speed DSP processor, the ADSP-2183 from Analog Devices, where the DXP-4C used an NEC uPD77016 processor. It is thus not firmware compatible. All algorithms which have been developed for the DXP-4C will be ported to the new product. However, due to memory limitations, new algorithms developed for the DXP-2X may not be ported to the DXP-4C.
- The DXP-2X uses a larger capacity FPGA for digital filtering, which allows peaking times of up to 40 μsec, with higher precision.
- The processor within the new DXP-2X can exchange data through the IDMA port with the CAMAC host while acquiring data, whereas the old DXP-4C had to be halted first. The IDMA transfer rate can support Level 1 FastCamac transfers (up to 5Mbyte/sec as opposed to 2MB/sec in the DXP-4C) which will be useful for large systems.

- The new DXP-2X can have up to 1 Mbyte/channel additional memory for special purposes, where the processor on the DXP-4C had a limited external memory space (32Kbyte/channel).
- The DXP-2X uses a 12 bit ADC, which can be operated up to 40 MHz sampling rate, which gives both larger dynamic range and better linearity than that in the DXP-4C.
- The analog section on the new DXP-2X is substantially lower in noise than the DXP-4C, and achieves Fe-55 energy resolution of better than 130 eV FWHM with a high resolution Ge detector. This also allows improved performance in the soft x-ray region (150-1000 eV).
- The internal gain control on the DXP-2X uses a single 16 bit DAC, which is simpler and more precise than the equivalent control on the DXP-4C.
- Both versions accept signals from either reset or resistive feedback preamplifiers. Model DXP-2X can accept a reset range of ± 10 volts (20 volt total range) without modification where the DXP-4C accepts only a 6 volt range.
- The analog inputs on the DXP-2X use SMA connectors, which are more reliable than the LEMO connectors on the DXP-4C. If the detector outputs use BNC cables, XIA can supply BNC to SMA adapters.
- The external Gate and Sync front panel signals use either NIM or TTL logic levels on the DXP-2X (jumper selectable), where they used TTL level signals on the DXP-4C. These digital signals still use LEMO connectors.

1.3. Module Specifications:

CAMAC Commands:

The supported CAMAC commands are described in Appendix B of this User's Manual.

Performance:

Energy Scale Integral Nonlinearity:	Less than 0.1% of full scale.
Peak stability with count rate:	Less than 0.1% up to highest counting rates at 5900 eV.
Resolution stability:	Less than 10% change up to maximum throughput. (For certain detectors, particularly POR, at very high event rates the resolution and non linearity may degrade faster, due to detector or preamplifier shifts).
Gain stability with temperature:	Less than 0.01%/degree C.
Temperature Range:	0° C - 50° C
Cooling air flow required:	200 ft/minute at 20° C, rising to 800 ft/minute at 50° C.

Power Requirements:

A four channel DXP module uses the following CAMAC voltage sources:

+6 volts	1.5 A	(9 watts) [to be verified]
-6 volts	0.5 A	(3 watts) [to be verified]
+12 volts	200 mA	(2.4 watts) [to be verified]
-12 volts	200 mA	(2.4 watts) [to be verified]

Note that unlike the predecessor DXP-4C, the DXP-2X uses the ±12 volt supplies, which are optional in the CAMAC standard. It is important to ensure that the CAMAC crate has these supplies, as many do not. The DXP can be used with inexpensive, portable CAMAC crates which have switching supplies, although energy resolution may degrade somewhat. To avoid ground loops, it is best to supply preamplifier power from the same CAMAC crate supplies that power the DXP's. The XIA CAMAC module PDM is recommended for this purpose; the PDM can supply power to 2 groups of up to 10 preamplifiers each on industry standard DB-9 connectors. Alternatively, it can be used in conjunction with the Model PBB-20 break-out box to supply power to up to 20 individual DB-9 connectors.

Warranties and Support:

The DXP hardware is warranted against all defects for 1 year. Please contact the factory or your distributor before returning items for service. If needed, XIA will attempt to provide "loaner" modules .

1.4 Introduction to the DXP:

The DXP can accommodate most common x-ray detector preamplifiers, and is especially well suited for single or multi-element Si(Li) and germanium detectors with reset preamplifiers. A typical multi-element detector array system equipped with DXP readout is shown in Figure 1.1.

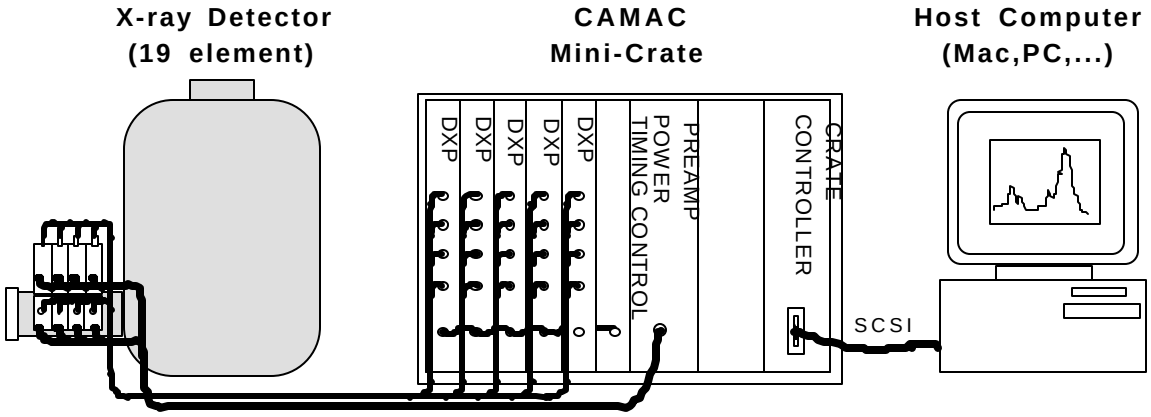


Figure 1.1: Schematic of a 19 element x-ray detector read out using 5 DXP modules. Each preamplifier has one input signal into a DXP channel. The preamplifiers can all be powered from the same CAMAC crate. A timing control module is shown to synchronize the data taking. The host processor can control the CAMAC system via numerous methods.

The DXP will accommodate either positive or negative polarity input signals, software selectable. When used with a reset preamplifier, the DXP-2X analog input accepts detector signals with a reset range between –10 volts and +10 volts. It is possible to accommodate input signals outside that range by reducing the input stage gain by a factor of four with a jumper.

Each DXP board has 4 channels and so can accommodate up to four preamplifier channels per module (the DXP-2X is also sold in a 2-channel version supporting 2 detector channels). Each channel consists of four basic sections, shown below in Figure 1.2: a front-end Analog Signal Conditioner (ASC); an ADC digitizing at 40 MHz; a digital Filter, Peak detector, Pileup Inspector (FiPPI) to filter the digitized signal stream and capture x-ray events; and a Digital Signal Processor (DSP) for pulse height analysis, data corrections, control of the other system sections (ASC & FiPPI), and communication with a host processor.

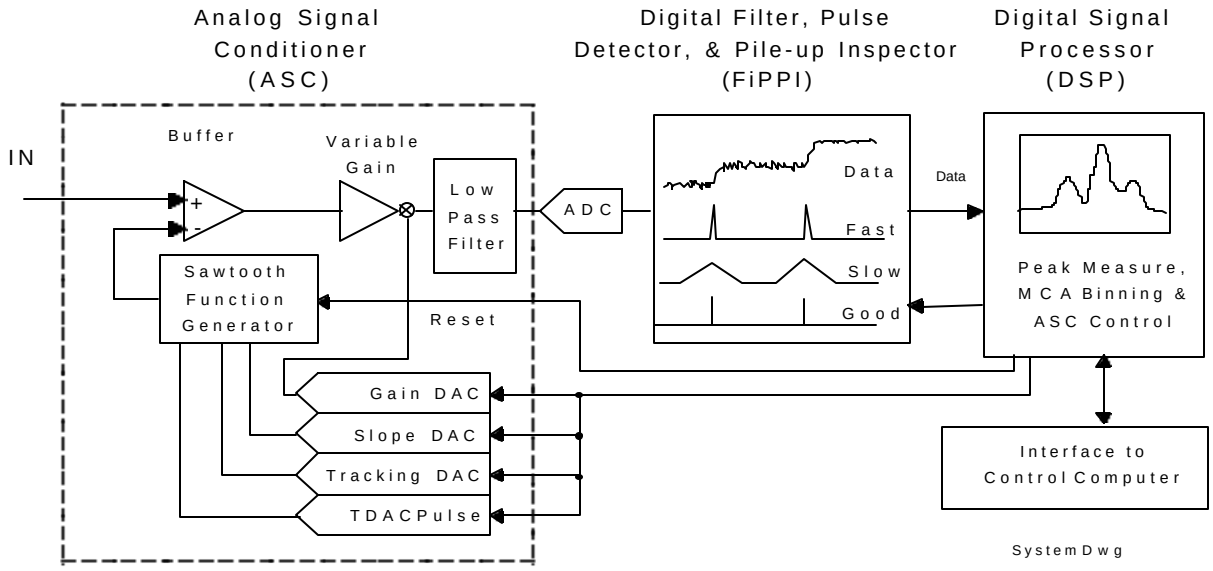


Figure 1.2: Block diagram of the DXP channel architecture, showing the major functional sections.

The preamplifier output signal feeds in to the ASC via a front panel SMA connector with 10 k□ impedance. The role of the ASC is to match the signals from a wide variety of commonly used preamplifiers to the range and sampling rate of the ADC. It does this by subtracting an offset signal, which in the case of reset preamplifiers is an internally generated ramp signal, and scaling the difference by a programmable gain. The DSP monitors the ASC's behavior, periodically adjusts the control DACs, and detects preamplifier resets to reset the internal ramp generator. Before being digitized the signal bandwidth is limited with a Butterworth low-pass filter to meet the Nyquist criterion.

The FiPPI utilizes a pair of trapezoidal filters: the "fast" filter, with a short peaking time for event selection and pile-up rejection; and the "slow" filter, with a longer peaking time for better energy resolution. [Note that a triangularly shaped pulse of peaking time t has approximately the same duration as a semi-Gaussian shaped pulse of shaping time t/2.] Both trapezoidal filters have programmable peaking times and gaps, where the gap (or "flat top") can be adjusted to compensate for preamp rise times (to avoid problems caused by "ballistic deficit"). The use of the fast filter and digital pile-up inspection decreases the dead-time per event to be just the pulse base-width (i.e. twice the peaking time + gap), which is less than the dead time for comparable analog systems. For the shortest peaking time (0.125 μsec) an output count rate of more than 500 kHz can be achieved. Fast filter parameters such as the discriminator threshold and pile-up rejection criteria are completely programmable. The FiPPI has been implemented in a Xilinx field programmable gate array (FPGA), and thus may be reprogrammed for special purposes.

The DSP, optimized for fixed point arithmetic and high I/O rate, applies data corrections to achieve optimal resolution with either pulsed optical reset or resistor feedback preamplifiers. While collecting data, the DSP continually monitors and controls the ASC output to match the ADC input range. The DSP operates at up to the highest event rates with very low dead-time from processor overhead. Other sources of dead-time (such as preamplifier resets) tend to be larger. In any case, the total live-time during data taking is accurately recorded. The total number of fast-peak triggers (i.e. the measured ICR) is also recorded to allow precise correction for dead-time due to pulse pile-up.

The standard firmware for the DSP processors allows full MCA acquisition for each channel, with up to 8k (8192) bins in each spectrum. This firmware can be customized for special purposes; interested persons should contact XIA for further information. In addition, the control software running on the host computer is an important part of the system. Several options exist for implementations on different platforms. A set of C and FORTRAN callable driver routines have been developed to assist those who wish to integrate DXP control into their existing data collection programs. These routines have been successfully integrated into

several available XAS control software packages including SPEC and XAS_COLLECT. XIA has also developed a custom software package specifically designed to work with multi-element detectors. MESA (Multi Element Spectrum Analysis) provides tools which aid in gain matching between channels as well as find the optimal settings for each peaking time. Please contact XIA or XIA's web site (www.xia.com) for further details on these and other options.

The DXP-2X Model C module has an external NIM or TTL gate signal to provide the option of controlling data acquisition from an external timing source, such as a CAMAC real time clock. The Model T has two additional inputs -- SYNC and AUX -- which may be used for various special purposes, as described in Appendix E. These include switching data collection between multiple spectra in the DSP synchronously with some external experimental parameter (e.g. phase locked EXAFS), and time resolved multi-channel scaling, including "quick-EXAFS". Either model can be equipped with up to 1 MByte of additional memory per channel, for larger spectra or other special applications.

This page intentionally blank.

2. Digital Filtering Theory, DXP Structure and Theory of Operation:

The purpose of this section is to provide the general DXP user with an explanation of its operation which is deep and complete enough to allow the module to be used effectively yet not so filled with detail as to become cumbersome. A further level of detail is required for those who wish to engage in developing control programs for the DXP and this is provided in the companion volumes *DSP Software Manual for the DXP 4C/4T Digital X-ray Processor* [Ref. 1] and *Host Software Description Manual for the DXP 4C/4T Digital X-ray Processor* [Ref. 3].

This introduction is divided into three sections. In the first, we examine the general issues associated with using a digital processor to extract accurate x-ray energies from a preamplifier signal and detect and eliminate pile-ups. In the second section we then describe how these general functions are specifically implemented in the DXP. This leads rather naturally to a discussion of the parameters used to control the DXP's functions: that is, those digital values which replace knob positions in analog systems. In the third section we the proceed to describe strategies both for selecting reasonable starting parameter values and for adjusting their values to optimize performance in particular situations.

2.1. X-ray Detection and Preamplifier Operation:

Energy dispersive detectors, which include such solid state detectors as Si(Li), HPGe, HgI_2, CdTe and CZT detectors, are generally operated with charge sensitive preamplifiers as shown in Figure 2.1a. Here the detector D is biased by voltage source V and connected to the input of amplifier A which has feedback capacitor C_f. In resetting preamplifiers a switch S is provided to short circuit C_f from time to time when the amplifier's output voltage gets so large that it behaves nonlinearly. Switch S may be an actual transistor switch, or may operate equivalently by another mechanism. In pulsed optical reset preamps light is shined on the amplifier A's input FET to cause it to discharge C_f. In PentaFET circuits, the input FET has an additional electrode which can be pulsed to discharge C_f.

The output of the preamplifier following the absorption of an x-ray of energy E_x in detector D is shown in Figure 2.1b as a step of amplitude V_x. When the x-ray is absorbed in the detector material it releases an electric charge $Q_x = E_x/\varepsilon$, where ε is a material constant. Q_x is integrated onto C_f, to produce the voltage $V_x = Q_x/C_f = E_x/(\varepsilon C_f)$. Measuring the energy E_x of the x-ray therefore requires a measurement of the voltage step V_x in the presence of the amplifier noise σ, as indicated in Fig. 2.1b.

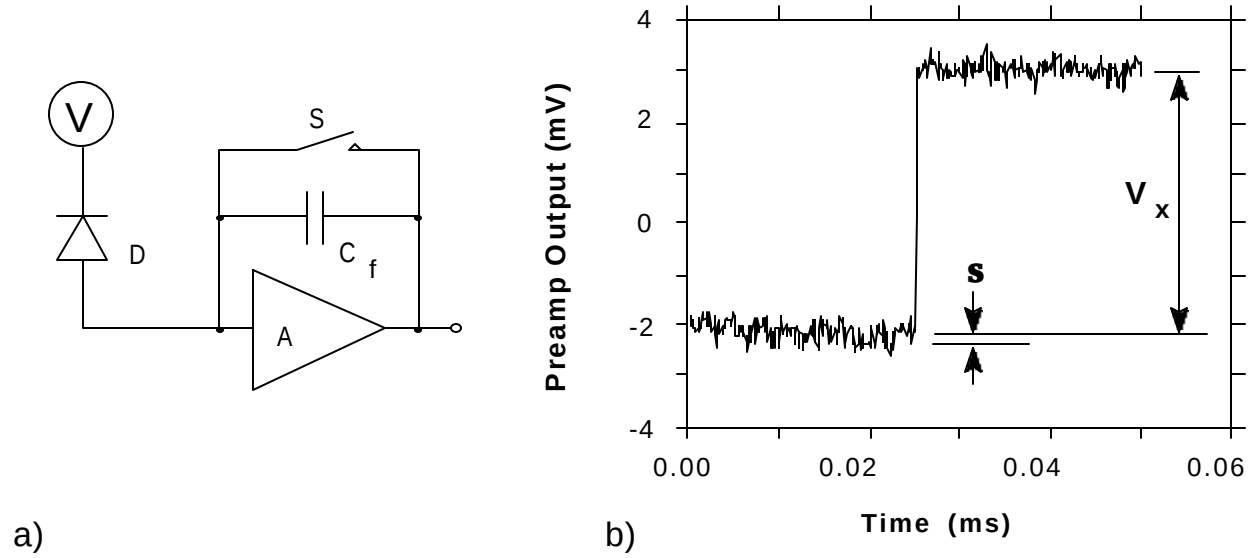


Figure 2.1: a) Charge sensitive preamplifier with reset; b) Output on absorption of an x-ray.

2.2. X-ray Energy Measurement & Noise Filtering:

Reducing noise in an electrical measurement is accomplished by filtering. Traditional analog filters use combinations of a differentiation stage and multiple integration stages to convert the preamp output steps, such as shown in Fig. 2.1b, into either triangular or semi-Gaussian pulses whose amplitudes (with respect to their baselines) are then proportional to V_x and thus to the x-ray's energy.

Digital filtering proceeds from a slightly different perspective. Here the signal has been digitized and is no longer continuous, but is instead a string of discrete values, such as shown in Figure 2.2. Fig. 2.2 is actually just a subset of Fig. 2.1b, which was dititized by a Tektronix 544 TDS digital oscilloscope at 10 MSA (megasamples/sec). Given this data set, and some kind of arithmetic processor, the obvious approach to determining V_x is to take some sort of average over the points before the step and subtract it from the value of the average over the points after the step. That is, as shown in Fig. 2.2, averages are computed over the two regions marked "Length" (the "Gap" region is omitted because the signal is changing rapidly here), and their difference taken as a measure of V_x. Thus the value V_x may be found from the equation:

$$V_{x,k} = -\sum_{i\,(before)} w_i v_i + \sum_{i\,(after)} w_i v_i \qquad (2.1)$$

where the values of the weighting constants w_i determine the type of average being computed. The sums of the values of the two sets of weights must be individually normalized.

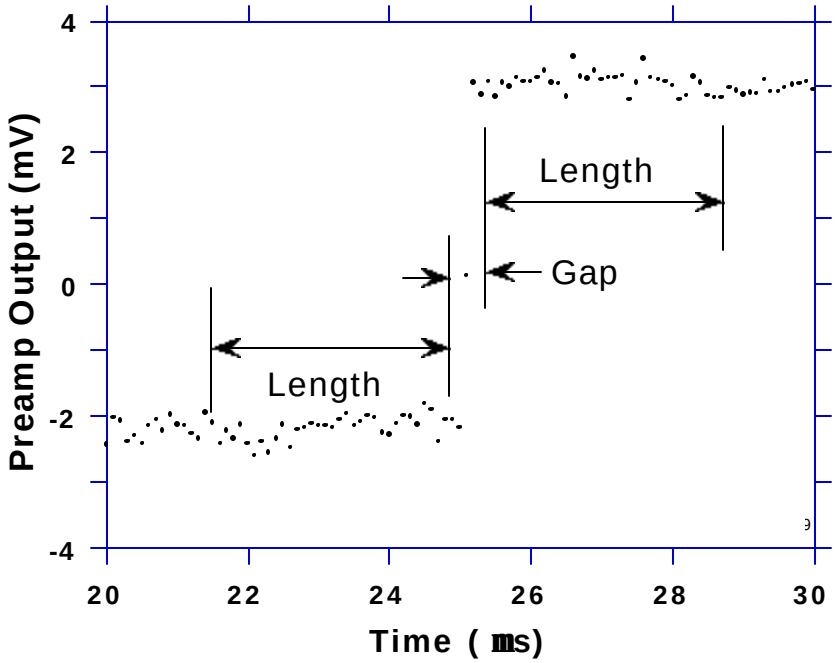


Figure 2.2: Digitized version of the data of Fig. 3B in the step region.

The primary differences between different digital signal processors lie in two areas: what set of weights $\{w_i\}$ is used and how the regions are selected for the computation of Eqn. 2.1. Thus, for example, when the weighting values decrease with separation from the step, then Eqn. 2.1 produces "cusp-like" filters. When the weighting values are constant, one obtains triangular (if the gap is zero) or trapezoidal filters. The concept behind cusp-like filters is that, since the points nearest the step carry the most information about its height, they should be most strongly weighted in the averaging process. How one chooses the filter lengths results in time variant (the lengths vary from pulse to pulse) or time invariant (the lengths are the same for all pulses) filters. Traditional analog filters are time invariant. The concept behind time variant filters is that, since the x-rays arrive randomly and the lengths between them vary accordingly, one can make maximum use of the available information by setting Length to the interpulse spacing.

In principal, the very best filtering is accomplished by using cusp-like weights and time variant filter length selection. There are serious costs associated with this approach however, both in terms of computational power required to evaluate the sums in real time and in the complexity of the electronics required to generate (usually from stored coefficients) normalized $\{w_i\}$ sets on a pulse by pulse basis. A few such systems have been produced but typically cost about $13K per channel and are count rate limited to about 30 Kcps. Even time invariant systems with cusp-like filters are still expensive due to the computational power required to rapidly execute strings of multiply and adds. One commercial system exists which can process over 100 Kcps, but it too costs over $12K per channel.

The DXP processing system developed by XIA takes a different approach because it was optimized for very high speed operation and low cost per channel. It implements a fixed length filter with all w_i values equal to unity and in fact computes this sum afresh for each new signal value k. Thus the equation implemented is:

$$L\,V_{x,k} = -\sum_{i=k-2L-G+1}^{k-L-G} v_i + \sum_{i=k-L+1}^{k} v_i, \qquad (2.2)$$

where the filter length is L and the gap is G. The factor L multiplying $V_{x,k}$ arises because the sum of the weights here is not normalized. Accommodating this factor is trivial for the DXP's host software. In the DXP, Eqn. 2.2 is actually implemented in hardwired logic by noting the recursion relationship between $V_{x,k}$ and $V_{x,k-1}$, which is:

$$L\,V_{x,k} = L\,V_{x,k-1} + v_k - v_{k-L} - v_{k-L-G} + v_{k-2L-G} \qquad (2.3)$$

While this relationship is very simple, it is still very effective. In the first place, this is the digital equivalent of triangular (or trapezoidal if G ð 0) filtering which is the analog industry's standard for high rate processing. In the second place, one can show theoretically that if the noise in the signal is white (i.e. Gaussian distributed) above and below the step, which is typically the case for the short shaping times used for high signal rate processing, then the average in Eqn. 2.2 actually gives the best estimate of V_x in the least squares sense. This, of course, is why triangular filtering has been preferred at high rates. Triangular filtering with time variant filter lengths can, in principle, achieve both somewhat superior resolution and higher throughputs but comes at the cost of a significantly more complex circuit and a rate dependent resolution, which is unacceptable for many types of precise analysis. In practice, XIA's design has been found to duplicate the energy resolution of the best analog shapers while approximately doubling their throughput, providing experimental confirmation of the validity of the approach.

2.3. Trapezoidal Filtering in the DXP:

From this point onward, we will only consider trapezoidal filtering as it is implemented in the DXP according to Eqns. 2.2 and 2.3. The result of applying such a filter with Length L = 20 and Gap G = 4 to the same data set of Fig. 2.2 is shown in Figure 2.3. The filter output V_x is clearly trapezoidal in shape and has a risetime equal to L, a flattop equal to G, and a symmetrical falltime equal to L. The basewidth, which is a first-order measure of the filter's noise reduction properties, is thus 2L+G. This raises several important points in comparing the noise performance of the DXP to analog filtering amplifiers. First, semi-Gaussian filters are usually specified by a *shaping time*. Their peaking time is typically twice this and their pulses are not symmetric so that the basewidth is about 5.6 times the shaping time or 2.8 times their peaking time. Thus a semi-Gaussian filter typically has a slightly better energy resolution than a triangular filter of the same peaking time because it has a longer filtering time. This is typically accommodated in amplifiers offering both triangular and semi-Gaussian filtering by stretching the triangular peaking time a bit, so that the *true* triangular peaking time is typically 1.2 times the selected semi-Gaussian peaking time. This also leads to an apparent advantage for the analog system when its energy resolution is compared to a digital system with the same nominal peaking time.

One extremely important characteristic of a digitally shaped trapezoidal pulse is its extremely sharp termination on completion of the basewidth 2L+G. This may be compared to analog filtered pulses which

have tails which may persist up to 40% of the peaking time, a phenomenon due to the finite bandwidth of the analog filter. As we shall see below, this sharp termination gives the digital filter a definite rate advantage in pileup free throughput.

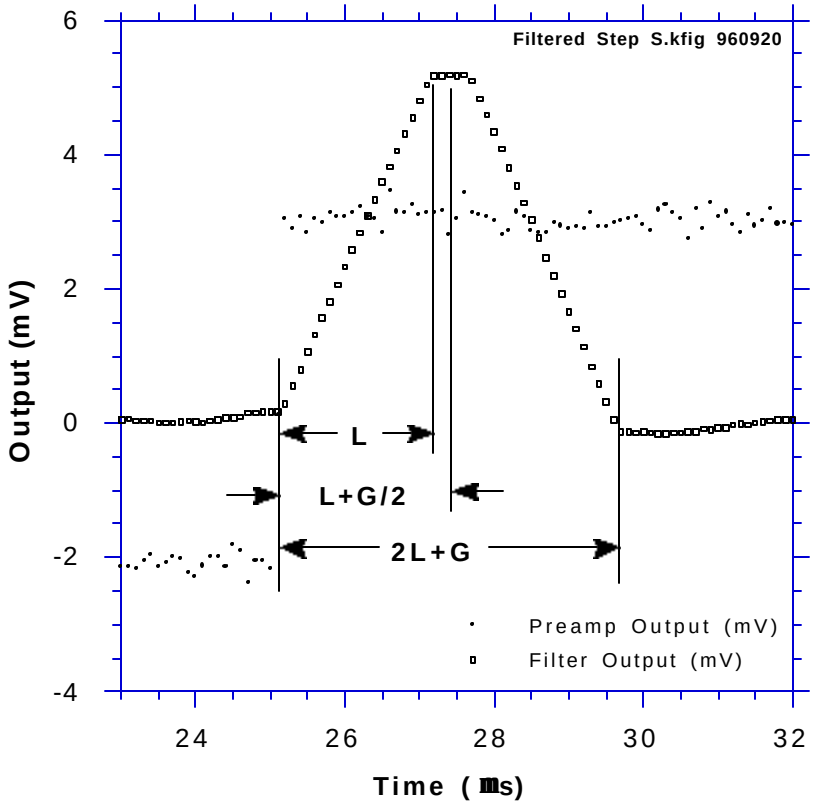


Figure 2.3: Trapezoidal filtering the Preamp Output data of Fig. 2.2 with L = 20 and G = 4.

2.4. Baseline Issues:

Figure 2.4 shows the same event as is Fig. 2.3 but over a longer time interval to show how the filter treats the preamplifier noise in regions when no x-ray pulses are present. As may be seen the effect of the filter is both to reduce the amplitude of the fluctuations and reduce their high frequency content. This signal is termed the *baseline* because it establishes the reference level from which the x-ray peak amplitude V_X is to be measured. The fluctuations in the baseline have a standard deviation σ_e which is referred to as the *electronic noise* of the system, a number which depends on the peaking time of the filter used. Riding on top of this noise, the x-ray peaks contribute an additional noise term, the *Fano noise*, which arises from statistical fluctuations in the amount of charge Q_X produced when the x-ray is absorbed in the detector. This Fano noise σ_f adds in quadrature with the electronic noise, so that the total noise σ_t in measuring V_X is found from

$$\sigma_t = \text{sqrt}(\sigma_f^2 + \sigma_e^2). \qquad (2.4)$$

The Fano noise is only a property of the detector material. The electronic noise, on the other hand, may have contributions from both the preamplifier and the amplifier. When the preamplifier and amplifier are both well designed and well matched, however, the amplifier's noise contribution should be essentially negligible. Achieving this in the mixed analog-digital environment of a digital pulse processor is a non-trivial task, however.

In the general case, however, the mean baseline value in not zero. This situation arises whenever the slope of the preamplifier signal it not zero between x-ray pulses. This can be seen from Eqn. 2.2. When the slope is not zero, the mean values of the two sums will differ because they are taken over regions separated in time by L+G, on average. Such non-zero slopes can arise from various causes, of which the most common is detector leakage current.

When the mean baseline value is not zero, it must be determined and subtracted from measured peak values in order to determine V_x values accurately. If the error introduced by this subtraction is not to significantly increase σ_t, then the error in the baseline estimate σ_b must be small compared to σ_e. Because the error in a single baseline measurement will be σ_e, this means that multiple baseline measurements will have to be averaged. In the standard DXP operating code this number is 64, which leads to the total noise shown in Eqn. 2.5.

$$\sigma_t = \mathrm{sqrt}(\sigma_f^2 + (1+1/64)\sigma_e^2). \qquad (2.5)$$

This results in less than 0.5 eV degradation in resolution even for very long peaking times when resolutions of order 140 eV are obtained.

In practice, the DXP initially makes a series of 64 baseline measurements to compute a starting baseline mean. It then makes additional baseline measurements at quasi-periodic intervals to keep the estimate up to date. These values are stored internally and can be read out to construct a spectrum of baseline noise. This is recommended because of its excellent diagnostic properties. When all components in the spectrometer system are working properly, the baseline spectrum should be Gaussian in shape with a standard deviation reflecting σ_n. Deviations from this shape indicate various pathological conditions which also cause the x-ray spectrum to be distorted and which should be fixed.

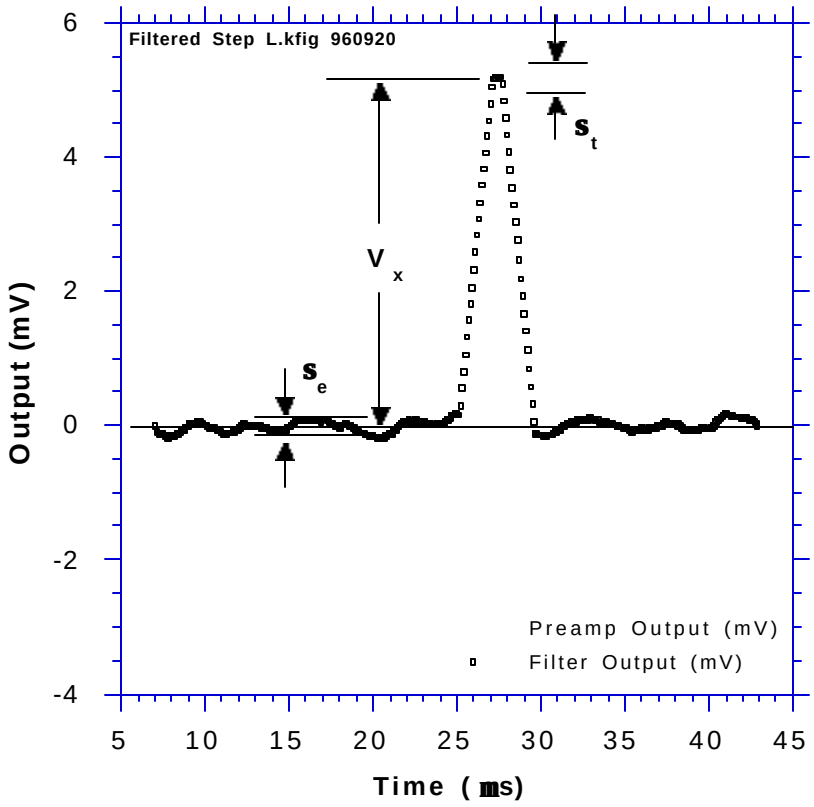


Figure 2.4: The event of Fig. 2.3 displayed over a longer time period to show baseline noise.

2.5. X-ray Detection & Threshold Setting:

As noted above, we wish to capture a value of V_X for each x-ray detected and use these values to construct a spectrum. This process is also significantly different between digital and analog systems. In the analog system the peak value must be "captured" into an analog storage device, usually a capacitor, and "held" until it is digitized. Then the digital value is used to update a memory location to build the desired spectrum. During this analog to digital conversion process the system is dead to other events, which can severely reduce system throughput. Even single channel analyzer systems introduce significant deadtime at this stage since they must wait some period (typically a few microseconds) to determine whether or not the window condition is satisfied.

Digital systems are much more efficient in this regard, since the values output by the filter are already digital values. All that is required is to capture the peak value – it is immediately ready to be added to the spectrum. If the addition process can be done in less than one peaking time, which is usually trivial digitally, then no system deadtime is produced by the capture and store operation. This is a significant source of the enhanced throughput found in digital systems.

In the DXP the peak detection and sampling is handled as indicated in Figure 2.5. In the DXP two trapezoidal filters are implemented, a *fast filter* and a *slow filter*. The fast filter is used to detect the arrival of x-rays, the slow filter is used to reduce the noise in the measurement of V_X, as described in the sections above. Fig. 2.5 shows the same data as in Figs. 2.1 - 2.4, together with the normalized fast and slow filter outputs. The fast filter has a filter length $L_f = 4$ and a gap $G_f = 0$. The slow filter has $L_S = 20$ and $G_S = 4$. Because the samples were taken at 10 MSA, these correspond to peaking times of 400 ns and 2 µs, respectively.

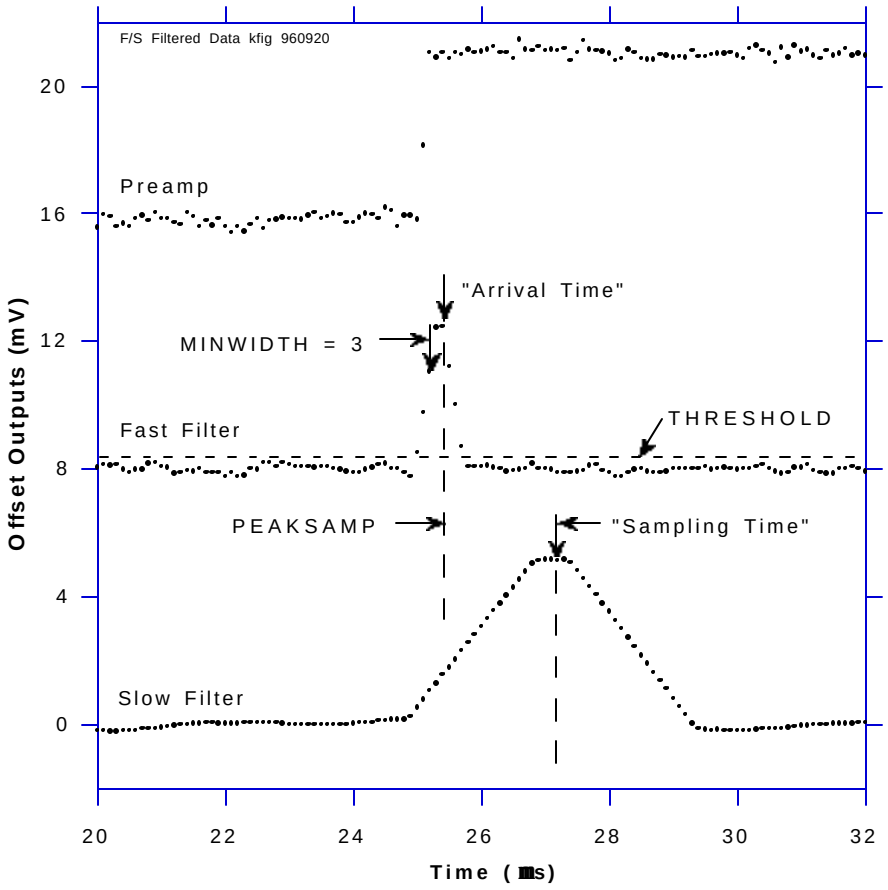


Figure 2.5: Peak detection and sampling methods in the DXP digital processor.

The arrival of the x-ray step (in the preamp output) is detected by digitally comparing the fast filter output to the digital constant THRESHOLD, which represent a threshold value. Once the threshold is exceeded, the number of values above threshold are counted. If they exceed a minimum number MINWIDTH, then the excursion is classified as a true peak and not a noise fluctuation. This scheme is much more noise resistant than a simple discriminator circuit, which triggers anytime the threshold is crossed. Thus THRESHOLD can be set much closer to the noise floor, which can be particularly advantageous when working with low energy x-rays. Once the MINWIDTH criterion has been satisfied, the DXP finds the arrival of the largest value (which becomes the pulse's official "arrival time") and starts a counter to count PEAKSAMP clock cycles to arrive at the appropriate time to sample the value of the slow filter. Because the digital filtering processes are deterministic, PEAKSAMP depends only on the values of the fast and slow filter constants and the risetime of the preamplifier pulses. The slow filter value captured following PEAKSAMP is then the slow digital filter's estimate of V_X.

2.6. Energy Measurement with Resistive Feedback Preamplifiers

In previous sections, the pulse height measurement was shown for the case of pulsed reset preamplifiers. The pulsed reset scheme is most often used for optimum energy resolution x-ray detectors. Other detectors use a continuous reset which we refer to as "resistive feedback" or "RC feedback", where the reset switch S in Figure 2.1 is replaced by a large value resistor, giving a exponential decay time of typically 50 μsec. The RC feedback type preamplifier is most often used for gamma-ray detectors which cover a larger dynamic range and where the electronic noise is not as significant a contribution to energy resolution.

Where analog shaping amplifiers typically have a "pole-zero" adjustment to cancel out the exponential decay, the DXP uses a patented exponential decay correction to achieve good energy resolution without a pole-zero correction. Figures 3.6 and 3.7 illustrate the method used. The first shows the output voltage of a RC feedback preamplifier with a x-ray or γ-ray step of amplitude A appearing at t=0. V_e is the voltage just before the step pulse arrives and V_0 is the asymptotic value that the signal would decay to in the absence of steps. t_1 is the earliest time used in the slow filter, L and G are the length and gap of the trapezoidal filter in clock units, and Δt is the clock period, In addition to the slow filter measurement, the ADC amplitude, V_D is made at time t_D. In the following discussion, it is assumed that the signal rise-time is negligible.

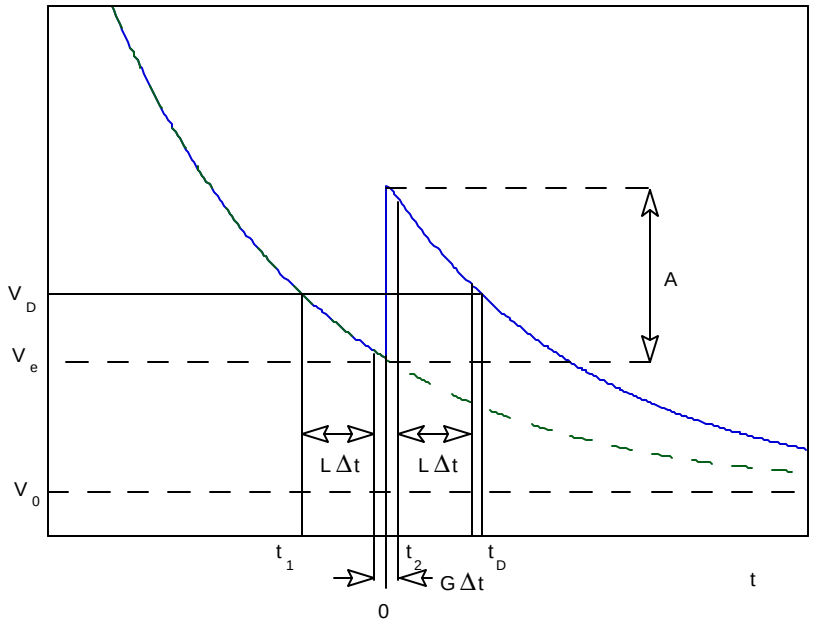


Figure 2.6: RC preamplifier output voltage. An x-ray step occurs at time t=0.

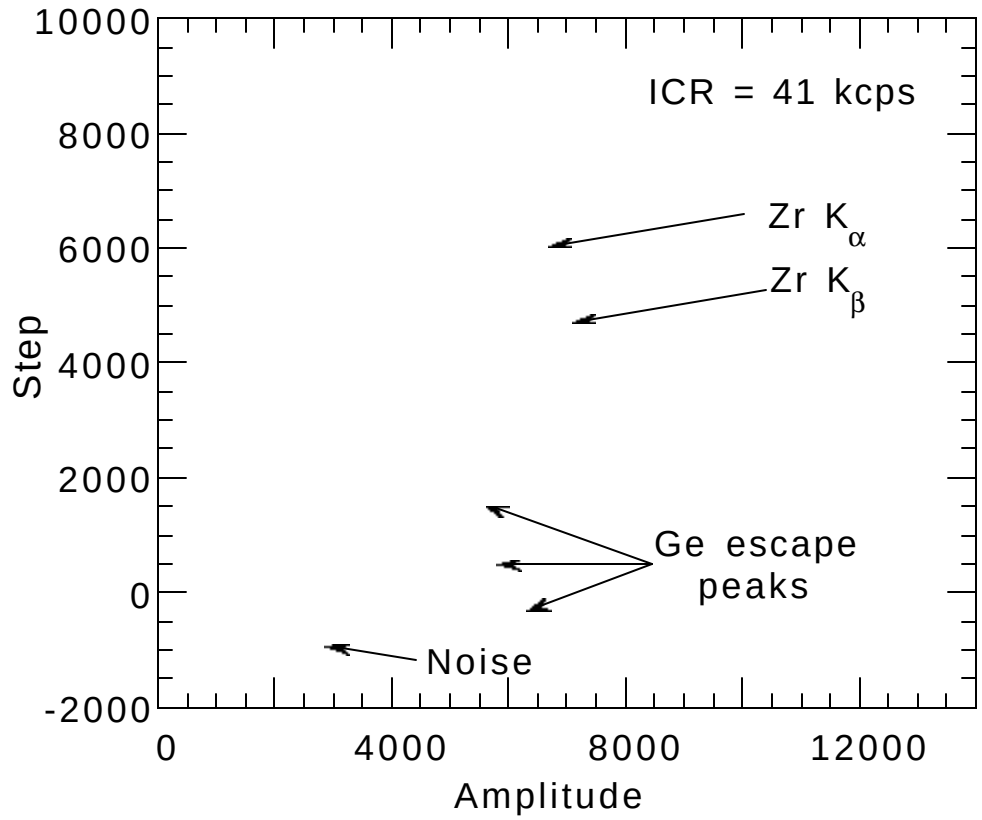


Figure 2.7: Correlation between step size and amplitude for Zr K_α x-ray events measured with the DXP-4C.

As Figure 2.7 makes clear, there is a linear correlation between the step height from the trapezoidal filter and the ADC amplitude, for pulses of a given energy. This is due to the fact that the exponential decay causes a deficit in the measured step height, which grows linearly with the distance from the asymptotic ADC offset at zero count rate.

The DSP reads these two values for each event that passes the FiPPI's trigger criteria, and makes a correction of the form:

$$E = k_1 (S_X + k_2 V_X - < S_B + k_2 V_B >)$$

Here the quantities S_X and V_X are the step height and ADC amplitude measured for the step, and the corresponding values with the B subscript are "baseline" values, which are measured frequently at times when there is no trigger. The brackets <> indicate that the baseline values are averaged over a large enough number of events to not introduce additional noise in the measurement. The constant k_2 (the DSP parameter called RCFCOR) is inversely proportional to the exponential decay time; this correction factor is a constant for a detector channel at a fixed gain and shaping time. The constant k_1 is effectively a gain factor, and is taken into account with a detector gain calibration.

The parameter RCFCOR is a function of the digital filter parameters (SLOWLEN, SLOWGAP and DECIMATION) and the preamplifier decay time (the DSP parameter TAURC). The decay time TAURC is in units of 50 ns clock ticks, and is measured with an exponential fit (for example, using the program DxpRCSetup). At the start of an acquisition run, the DSP calculates RCFCOR using the following approximate expression:

$$RCFCOR = 2^{DEC} * (LEN + GAP) / (TAURC - (LEN + GAP/2 + 3)*2^{DEC})$$

The above expression is valid for peaking times less than about TAURC/2. Alternatively, RCFCOR can be determined empirically in a special test run from a linear fit of data as in Figure 2.7.

2.7. Pile-up Inspection:

The value V_X captured at time PEAKSAMP after the x-ray pulse's arrival time will only be a valid measure of the associated x-ray's energy provided that the filtered pulse is sufficiently well separated in time from its preceding and succeeding neighbor pulses so that their peak amplitudes are not distorted by the action of the trapezoidal filter. That is, if the pulse is not *piled up*. The relevant issues may be understood by reference to Figure 2.6, which shows 5 x-rays arriving separated by various intervals.

Because the triangular filter is a linear filter, its output for a series of pulses is the linear sum of its outputs for the individual members in the series. In Fig. 2.6 the pulses are separated by intervals of 3.2, 1.8, 5.7, and 0.7 μs, respectively. The fast filter has a peaking time of 0.4 μs with no gap. The slow filter has a peaking time of 2.0 μs with a gap of 0.4 μs.

The first kind of pileup is *slow pileup*, which refers to pileup in the slow channel. This occurs when the rising (or falling) edge of one pulse lies under the peak (specifically the sampling point) of its neighbor. Thus, in Fig. 2.6, peaks 1 and 2 are sufficiently well separated so that the leading edge (point 2a) of peak 2 falls after the peak of pulse 1. Because the trapezoidal filter function is symmetrical, this also means that pulse 1's trailing edge (point 1c) also does not fall under the peak of pulse 2. For this to be true, the two pulses must be separated by at least an interval of L + G/2. Peaks 2 and 3, which are separated by only 1.8 μs, are thus seen to pileup in the present example with a 2.0 μs peaking time.

This leads to an important first point: whether pulses suffer slow pileup depends critically on the peaking time of the filter being used. The amount of pileup which occurs at a given average signal rate will increase with longer peaking times. We will quantify this in §2.6.

Because the fast filter peaking time is only 0.4 μs, these x-ray pulses do not pileup in the fast filter channel. The DXP can therefore test for slow channel pileup by measuring for the interval PEAKINT after a pulse arrival time. If no second pulse occurs in this interval, then there is no trailing edge pileup. PEAKINT is usually set to a value close to L + G/2 + 1. Pulse 1 passes this test, as shown in Fig. 2.6. Pulse 2, however, fails the PEAKINT test because pulse 3 follows in 1.8 μs, which is less than PEAKINT = 2.3 μs. Notice, by the symmetry of the trapezoidal filter, if pulse 2 is rejected because of pulse 3, then pulse 3 is similarly rejected because of pulse 2.

Pulses 4 and 5 are so close together that the output of the fast filter does not fall below the threshold between them and so they are detected by the pulse detector as only being a single x-ray pulse. Indeed, only a single (though somewhat distorted) pulse emerges from the slow filter, but its peak amplitude corresponds to the energy of neither x-ray 4 nor x-ray 5. In order to reject as many of these fast channel pileup cases as possible, the DXP implements a fast channel pileup inspection test as well.

The fast channel pileup test is based on the observation that, to the extent that the risetime of the preamplifier pulses is independent of the x-rays' energies (which is generally the case in x-ray work except for some room temperature, compound semiconductor detectors) the basewidth of the fast digital filter (i.e. $2L_f + G_f$) will also be energy independent and will never exceed some maximum width MAXWIDTH. Thus, if the width of the fast filter output pulses is measured at threshold and found to exceed MAXWIDTH, then fast channel pileup must have occurred. This is shown graphically in Fig. 2.8, where pulse 3 passes the MAXWIDTH test, while the piled up pair of pulses 4 and 5 fail the MAXWIDTH test.

Thus, in Fig. 2.8, only pulse 1 passes both pileup inspection tests and, indeed, it is the only pulse to have a well defined flattop region at time PEAKSAMP in the slow filter output.

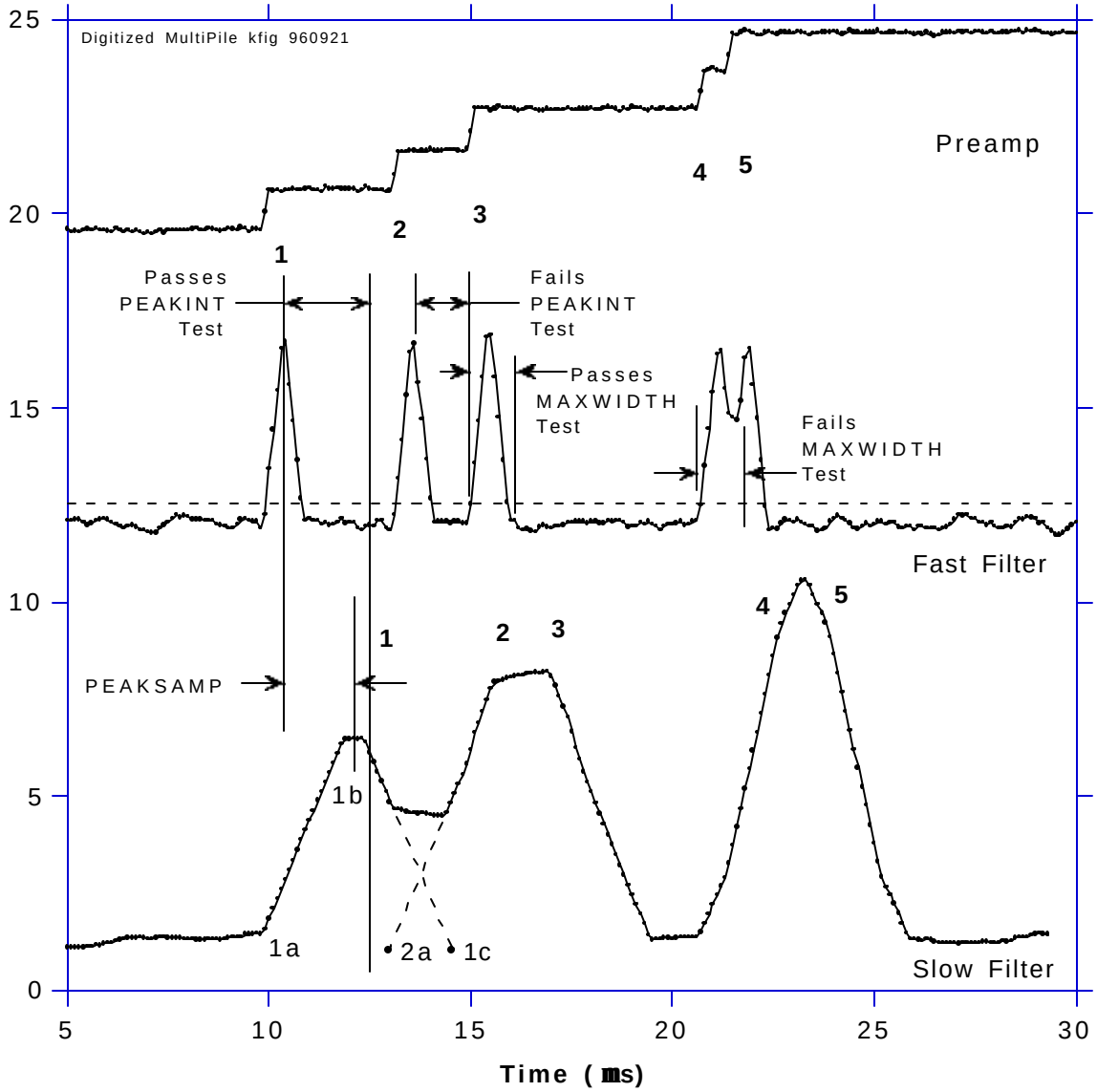


Figure 2.8: A sequence of 5 x-ray pulses separated by various intervals to show the origin of both slow channel and fast channel pileup and demonstrate how the two cases are detected by the DXP.

2.8. Input Count Rate (ICR) and Output Count Rate (OCR):

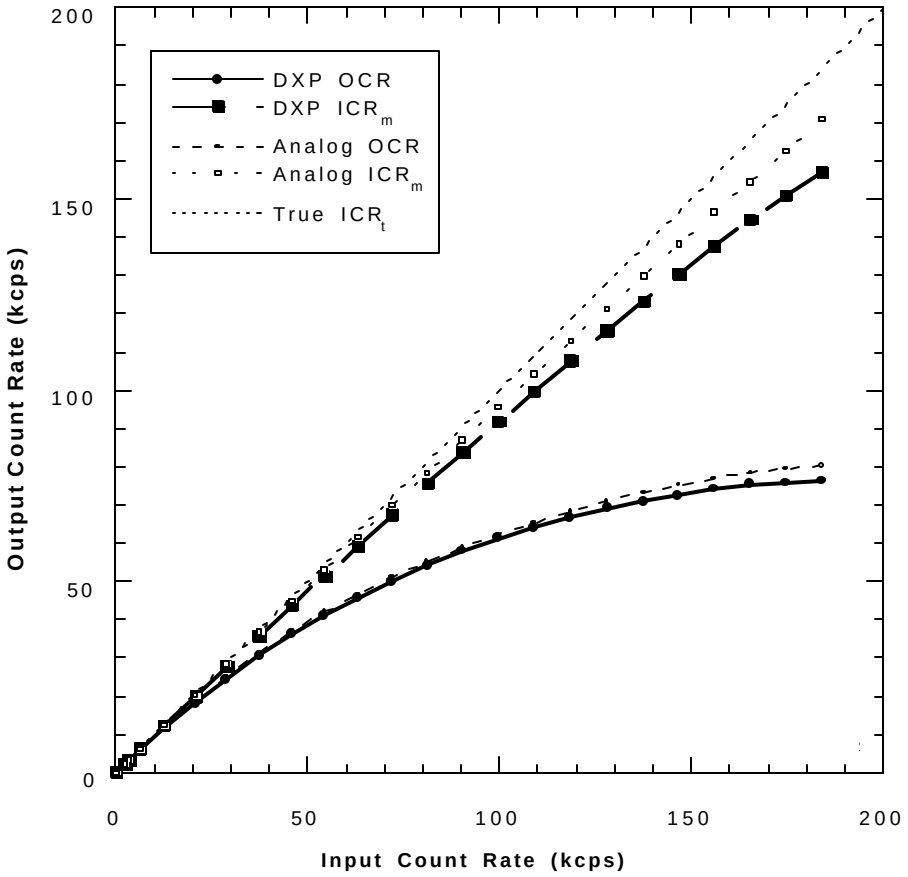
During data acquisition, x-rays will be absorbed in the detector at some rate. This is the *true input count rate* , which we will refer to as ICR_t. Because of fast channel pileup, not all of these will be detected by the DXP's x-ray pulse detection circuitry, which will thus report a *measured input count rate* ICR_m which will be less than ICR_t. This phenomenon, it should be noted, is a characteristic of all x-ray detection circuits, whether analog or digital, and is not specific to the DXP.

Of the detected x-rays, some fraction will also satisfy both fast and slow channel pileup tests and have their values of V_X captured and placed into the spectrum. This number is the *output count rate* , which we refer to as the OCR. The DXP normally returns, in addition to the collected spectrum, the actual time LIVETIME for which data was collected, together with the number FASTPEAKS of fast peaks detected and the number of V_X captured events EVTSINRUN. From these values, both the OCR and ICR_m can be computed according to Equation 6. These values can then be used to make deadtime corrections as discussed in the next section.

$$ICR_m = FASTPEAKS/LIVETIME; \quad OCR = EVTSINRUN/LIVETIME. \tag{2.6}$$

2.9. Throughput:

Figure 2.9 shows how the values of ICR_m and OCR vary with true input count rate for the DXP and compare these results to those from a common analog shaping amplifier plus SCA system. The data were taken at a synchrotron source using a detector looking at a CuO target illuminated by x-rays slightly above the Cu K edge. Intensity was varied by scanning a pair of slits across the input x-ray beam so that its harmonic content remained constant with varying intensity.



System	OCR Deadtime (ms)	ICR Deadtime (ms)
DXP (2 ms τ_p, 0.6 μs τ_g)	4.73	0.83
Analog Triangular Filter Amp (τ_p = 1 ms)	4.47	0.40

Figure 2.9: Curves of ICR_m and OCR for the DXP using 2 μs peaking time, compared to a common analog SCA system using 1 μs peaking time.

Functionally, the OCR in both cases is seen to initially rise with increasing ICR and then saturate at higher ICR levels. The theoretical form, from Poisson statistics, for a channel which suffers from paralyzable (extending) dead time [Ref. 4], is given by:

$$OCR = ICR_t * \exp(- ICR_t \ \tau_d), \tag{2.7}$$

where τ_d is the *dead time*. Both the DXP and analog systems' OCRs are so describable, with the *slow channel dead times* τ_{ds} shown in the Table below Fig. 2.9. The measured ICR_m values for both the DXP and analog

systems are similarly describable, with the *fast channel dead times* τ_{df} as shown. The maximum value of OCR can be found by differentiating Eqn. 2.7 and setting the result to zero. This occurs when the value of the exponent is -1, i.e. when ICR_t equals $1/\tau_d$. At this point, the maximum OCR_{max} is 1/e the ICR, or

$$OCR_{max} = 1/(e\ \tau_d) = 0.37/\tau_d. \qquad (2.8)$$

These are general results and are very useful for estimating experimental data rates.

The Table illustrates a very important result for using the DXP: the slow channel deadtime is nearly the minimum theoretically possible, namely the pulse basewidth. For the shown example, the basewidth is 4.6 μs ($2L_s + G_s$) while the deadtime is 4.73 μs. The slight increase is because, as noted above, PEAKINT is always set slightly longer than $L_s - G_s/2$ to assure that pileup does not distort collected values of V_x.

The deadtime for the analog system, on the other hand is much larger. In fact, as shown, the throughput for the digital system is almost twice as high, since it attains the same throughput for a 2 μs peaking time as the analog system achieves for a 1 μs peaking time. The slower analog rate arises, as noted earlier both from the longer tails on the pulses from the analog triangular filter and on additional deadtime introduced by the operation of the SCA. In spectroscopy applications where the system can be profitably run at close to maximum throughput, then, a single DXP channel will then effectively count as rapidly as two analog channels.

2.10. Dead Time Corrections:

The fact that both OCR and ICR_m are describable by Eqn. 2.7 makes it possible to correct DXP spectra quite accurately for deadtime effects. Because deadtime losses are energy independent, the measured counts N_{mi} in any spectral channel i are related to the true number N_{ti} which would have been collected in the same channel i in the absence of deadtime effects by:

$$N_{ti} = N_{mi}\ \ ICR_t/OCR. \qquad (2.9)$$

Looking at Fig. 2.7, it is clear that a first order correction can be made by using ICR_m in Eqn. 2.9 instead of ICR_t, particularly for OCR values less than about 50% of the maximum OCR value. For a more accurate correction, the fast channel deadtime τ_{df} should be measured from a fit to the equation

$$ICR_m = ICT_t * \exp(-\ ICR_t\ \ \tau_{df}\ ). \qquad (2.10)$$

Then, for each recorded spectrum, the associated value of ICR_m is noted and Eqn. 2.10 inverted (there are simple numerical routines to do this for transcendental equations) to obtain ICR_t. Then the spectrum can be corrected on a channel by channel basis using Eqn. 2.9. In experiments with a DXP prototype, we found that, for a 4 μs peaking time (for which the maximum ICR is 125 kcps), we could correct the area of a reference peak to better than 0.5% between 1 and 120 kcps. The fact that the DXP provides highly accurate measurements of both LIVETIME and ICR_m therefore allows it to produce accurate spectral measurements over extremely wide ranges of input counting rates.

3. DXP Structure and Description of Operation:

3.1. Organizational Overview:

The DXP channel architecture is shown in Figure 4.1, showing the three major operating blocks in the DXP: the Analog Signal Conditioner (ASC), Digital Filter, Peak Detector, and Pileup Inspector (FiPPI), and Digital Signal Processor (DSP). Signal digitization occurs in the Analog-to-Digital converter (ADC), which lies between the ASC and the FiPPI. In the DXP-2X, the ADC is a 12 bit, 40 MSA device, which is currently being used as a very linear 10-bit, 40 MHz ADC. The functions of the major blocks are summarized below.

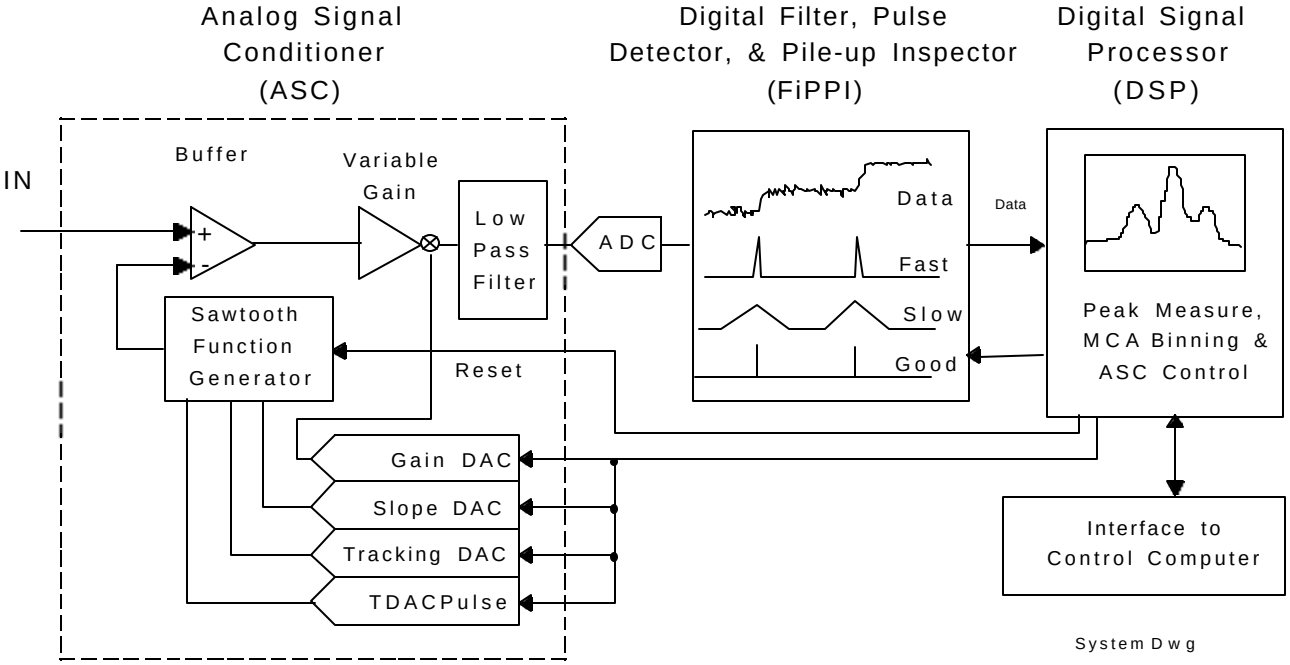


Figure 3.1: Block diagram of the DXP channel architecture, showing the major functional sections.

3.2. The Analog Signal Conditioner (ASC):

The ASC has two major functions: to reduce the dynamic range of the input signal so that it can be adequately digitized by a 10 bit converter and to reduce the bandwidth of the resultant signal to meet the Nyquist criterion for the following ADC. This criterion is that there should be no frequency component in the signal which exceeds half of the sampling frequency. Frequencies above this value are aliased into the digitized signal at lower frequencies where they are indistinguishable from original components at those frequencies. In particular, high frequency noise would appear as excess low frequency noise, spoiling the spectrometer's energy resolution. The DXP therefore has a 4 pole Butterworth filter with a cutoff frequency of about 10 MHz.

The dynamic range of the preamplifier output signal is reduced to allow the use of a 10 bit ADC, which greatly lowers the cost of the DXP. This need arises from two competing ADC requirements: speed and resolution. Speed is required to allow good pulse pileup detection, as described in §2.5. For high count rates, pulse pair resolution less than 200 ns is desirable, which implies a sampling rate of 10 MSA or more. The DXP uses a 40 MSA ADC. On the other hand, in order to reduce the noise σ in measuring V_X (see Fig. 2.1), experience shows that σ must be at least 4 times the ADC's single bit resolution ΔV_1. This effectively sets the gain of the amplifier stages preceding the ADC. Then, if the preamplifier's full scale voltage range is V_{max}, it must digitize to N bits, where N is given by:

$$N = \log_{10}(V_{max}/\Delta V1)/\log_{10}(2). \quad (11)$$

For a typical high resolution spectrometer, N must be 14 to 15. However, 14 bit ADCs operating in excess of 10 MSA are very expensive, particularly if their integral and differential non-linearities are less than 1 least significant bit (LSB). At the time of this writing a 10 bit 20 MSA ADC costs less than $10, while a 14 bit 5 MSA ADC costs nearly $500, which would more than triple the parts cost per channel.

The ASC circumvents this problem using a novel dynamic range technology, for which XIA has applied for a patent, which is indicated in Figure 3.2. Here a resetting preamplifier output is shown which cycles between about -3.0 and -0.5 volts. We observe that it is not the overall function which is of interest, but rather the individual steps, such as shown in Fig. 2.1b, which carry the x-ray amplitude information. Thus, if we know the average slope of the preamp output, we can generate a sawtooth function which has this average slope and restarts each time the preamplifier is reset, as shown in Fig. 3.2. If we then subtract this sawtooth from the preamplifier signal, we can amplify the difference signal to match the ADC's input range, also as indicated in the Figure. Gains of 8 to 16 are possible, thus reducing the required number of bits

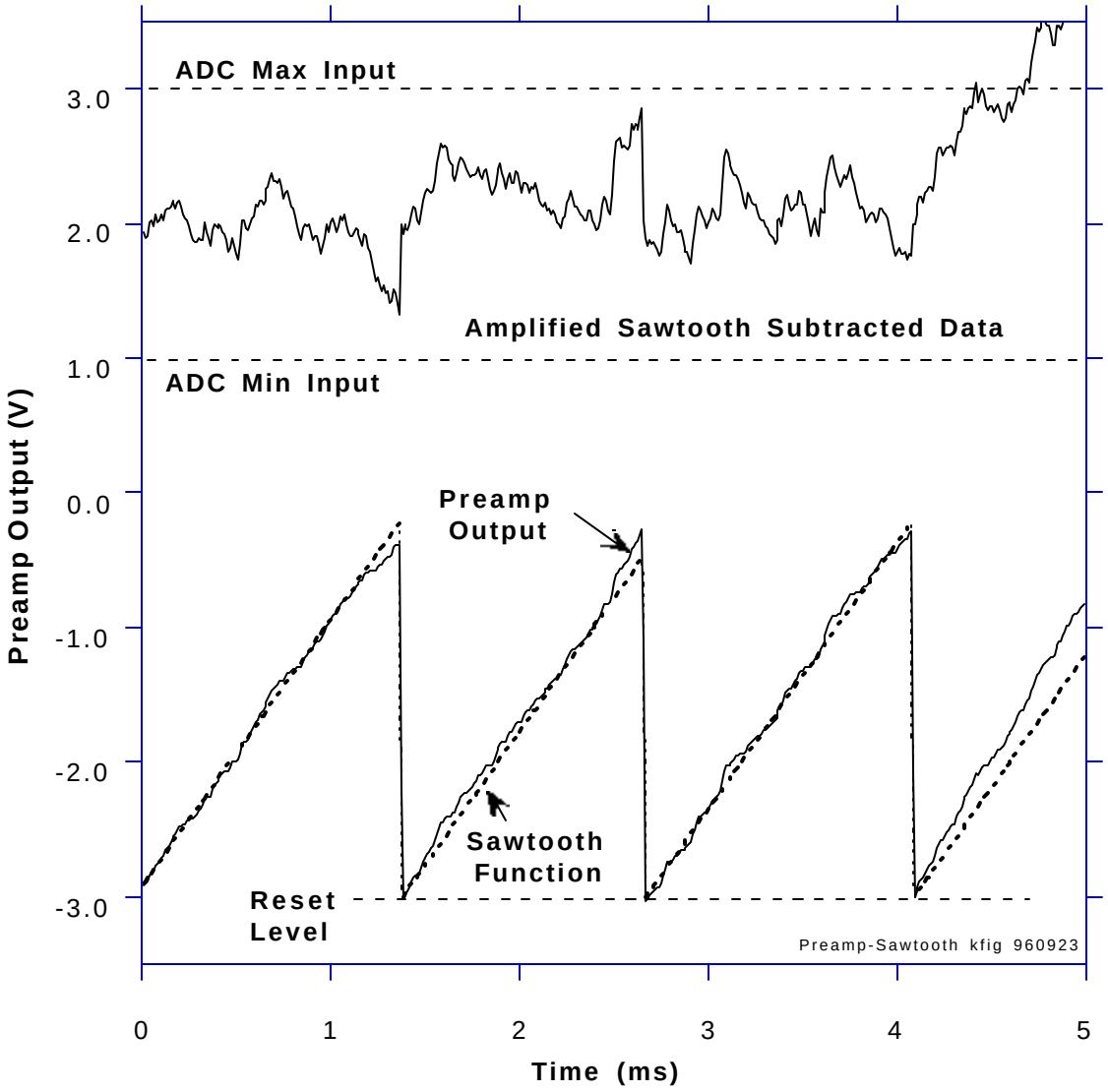


Figure 3.2: A sawtooth function having the same average slope as the preamp output is subtracted from it and the difference amplified and offset to match the input range of the ADC.

from 14 to 10. The generator required to produce this sawtooth function is quite simple, comprising a current integrator with an adjustable offset. The current, which sets the slope, is controlled by a DAC (SLOPEDAC), while the offset is controlled by a second DAC (TRACKDAC). The DAC input values are set by the DSP, which thereby gains the power to adjust the sawtooth generator in order to maintain the ASC output (i.e. the "Amplified Sawtooth Subtracted Data" of Fig. 3.2) within the ASC input range.

Occasionally, as also shown in Fig. 3.2, fluctuations in data arrival rate will cause the conditioned signal to pass outside the ADC input range. This condition is detected by the FiPPI, which has digital discrimination levels set to ADC zero and full scale, which then interrupts the DSP, demanding ASC attention. The DSP remedies the situation by adjusting TRACKDAC until the conditioned signal returns into the ADC's input range. During this time, data passed to the FiPPI are invalid. Preamplifier resets are detected similarly. When detected the DSP responded by setting TRACKDAC to a standard value corresponding the reset level (TRACKRST) and resetting the current integrator.

3.3. The Filter, Pulse Detector, & Pile-up Inspector (FiPPI):

The FiPPI is implemented in a field programmable gate array (FPGA) to accomplish the various filtering, pulse detection and pileup inspection tasks discussed in §2. As described there, it has a fast channel for pulse detection and pileup inspection and a slow channel for filtering, both with fully adjustable peaking times and gaps. The "fast" filter's τ_p (τ_{pf}) can be adjusted from 50 ns to 500 ns, while the "slow" filter's τ_p (τ_{ps}) can be adjusted from 0.125 µs to 40 µs. Adjusting τ_{pf} allows tradeoffs to be made between pulse pair resolution and the minimum x-ray energy that can be reliably detected. When τ_{pf} is 200 ns, for example, the pulse pair resolution is typically less than 200 ns. When τ_{pf} is 1 µs, x-rays with energies below 200 eV can be detected and inspected for pileup. To maximize throughput, τ_{ps} should be chosen to be as short as possible to meet energy resolution requirements, since the maximum throughput scales as $1/\tau_{ps}$, as per Eqn. 2.8. If the input signal displays a range of risetimes (as in the "ballistic deficit" phenomenon) the slow filter gap time can be extended to accommodate that range. The shortest value of τ_{ps} is 0.125 µs. At this setting, with a gap time of 100 ns, the dead time would be about 500 ns and the maximum throughput according to Eqn. 2.8 would be over 700 kcps.

The FiPPI also includes a livetime counter which counts the 40 MHz system clock, divided by 16, so that one "tick" is 400 ns. This counter is activated any time the DSP is enabled to collect x-ray pulse values from the FiPPI and therefore provides an extremely accurate measure of the system livetime. In particular, as described in §3.2, the DSP is not live either during preamplifier resets or during ASC out-of-ranges, both because it is adjusting the ASC and because the ADC inputs to the FiPPI are invalid. Thus the DXP measures livetime more accurately than an external clock, which is insensitive to resets and includes them as part of the total livetime. While the average number of resets/sec scales linearly with the countrate, in any given measurement period there will be fluctuations in the number of resets which may affect counting statistics in the most precise measurements.

All FiPPI parameters, including the filter peaking and gap times, threshold, and pileup inspection parameters are all externally supplied and may be adjusted by the user to optimize performance. Because the FiPPI is implemented in a Xilinx field programmable gate array (FPGA), it may be reprogrammed for special purposes, although this process is non-trivial and would probably require XIA contract support.

3.4. The Digital Signal Processor (DSP):

The Digital Signal Processor acquires and processes event data from the FiPPI, controls the ASC through DACs, and communicates with the host. The processor is an Analog Devices ADSP-2183 16 bit Fixed-Point DSP optimized for fixed point arithmetic and high I/O rates. Different DSP program variants are used for different types of data acquisition and different preamplifier types. Section ?? describes in detail the DSP operation, its tasks, and parameters which control them.

The ADSP-2183 has 16K words of 16-bit wide data memory and 16K words of 24-bit wide program memory, part of which is used as data memory to hold the MCA spectrum. (If more memory is required for special purposes, up to 1 Mbyte of extended memory can be added by specifying option M). Transferring data to/from these memory spaces is done through the DSP's built-in IDMA port, which does not interfere with the DSP program operation.

3.4.1. DSP Memory Organization:

The ADSP-2183 has 16k words of internal 24 bit program memory and 16K words of internal 16 bit data memory. The program memory holds the DSP code as well as the 8k MCA spectrum. Data memory contains all the DSP parameters (both externally visible and internally used), as well as space for a 1k baseline histogram and an additional data buffer used to hold baseline history data or adc trace data. The FiPPI control registers are mapped to external DSP memory space; see appendix D for details.

The address space for the ADSP-2183 is shown below:

0x0000 - 0x3FFF	DSP Program memory	Contains the DSP instructions and 24-bit MCA data
0x4000 - 0x7FDF	DSP Data memory	16-bit data, including the parameters memory
0x7FE0 - 0x7FFF	reserved	

3.4.2. Communications with the Host Computer:

Communications between the DSP and host computer occur via a CAMAC interface via the IDMA port on the DSP; through this port, the DSP internal memory looks like dual port memory, so the host can access the DSP memory space without affecting DSP operation. See appendix B for details on the actual CAMAC transfer.

The general DXP user does not have to understand these issues in detail and will instead use pre-existing, higher level software to interact with the module. XIA has developed software specifically designed to set up the DXP's for use with a multi-element detector system; *MESA* (for Multi Element Spectrum Analysis) is written using Labview from National Instruments, and runs on computers running Windows, Linux, or MacOS. XIA also supplies a set of C-code routines, callable from either C or FORTRAN, which can be used to simplify development efforts.

3.5. DSP Programs and Subprograms:

Different DSP code variants are employed for different purposes, being downloaded as required. Operation with reset and resistor feedback preamplifier, for example, require different variants because they require different function generator operation and different algorithms for computing x-ray energy from V_X values. At the time of this writing the following 5 variants exist: 1) Standard reset preamplifier; 2) Standard feedback preamplifier; 3) Additional diagnostic routines; 4) Switched dual MCA (data switching between two spectra under external SYNC control); and 5) Multichannel scaling (measures windowed counts vs time after external strobe pulse).

The DSP codes implement several subprograms which are required to setup and operate the DXP. These include: calibration tasks, which are executed as part of setting up the system; initial measurements, which are executed at the beginning of each data collection run; and data collection tasks, which are executed during the data collection run. Different code variants have different subsets of these subprograms, which are described briefly in the following subsections. More complete documentation is available in the DSP Software Manual [Ref. 1], which also explains how they are called.

3.5.1. Calibration Measurements:

All necessary calibrations are automatically performed by the DSP, either at startup time, or at the beginning of a run, if run conditions (gain setting, peaking time) have changed and recalibration is necessary. Unlike its predecessor, it is not necessary for the host to initiate any calibration runs.

3.5.2. Data Collection Tasks:

During data collection the DSP executes the following subprograms in addition to its V_X capture and x-ray energy computing and binning chores and adjusting the ASC to keep the signal within the ADC input range. These are as follows:

ASC Slope Monitoring: As noted in the discussion associated with Fig. 3.2 in §3.2, the ASC output signal (preamplifier minus sawtooth function) can go out of the ADC input range for various reasons including preamplifier resets and count rate fluctuations. If the sawtooth slope closely matches the average preamplifier ramp, then these fluctuations are equally likely to be positive or negative. If, however, the slope is incorrectly set, then there will be an excess of one or the other. In this task the DSP tracks the relative numbers of positive and negative excursions and, if there is an excess of one or the other, adjusts the slope generator control parameter SLOPEVAL accordingly.

Baseline Monitoring: Because the baseline value depends on rate and other factors which cannot be guaranteed to be constant during a data collection run, the DSP collects baseline values from time to time. It uses these values in two ways. First, it computes a running mean baseline value which is subtracted from V_X values in computing the x-ray energies. Secondly, it also produces a spectrum of all baseline values captured, using 512 channels of Y memory. This spectrum can be read out following the run and used for diagnostic purposes.

3.5.3. Diagnostic Tasks:

These tasks are not used in normal data collection procedures but are available in the diagnostic procedures variant to assist in system trouble shooting. All are described in further detail in the DSP Software Manual [Ref. 1], which also explains how they are called. The diagnostic tasks variant also contains all the subprograms listed in §3.5.1-3 above.

ADC Trace Measurement: Captures ADC traces directly, using the ADC as a digital oscilloscope (this is actually available in all code variants at present).

This page intentionally blank.

4. Initial DXP Setup With a New Preamplifier:

The DXP-2X has been designed to work well with most preamplifiers. For reset preamplifiers, the DXP-2X can accommodate a reset range anywhere between –10 volts and +10 volts (this range can be widened even further by reducing the input buffer gain by changing a jumper setting). Furthermore, the DXP-2X works well with preamplifier gains ranging from less than 1 mV/keV to over 10 mV/keV. In OFFSET mode, the DXP-2X also works well with feedback type preamplifiers.

4.1. Overview of the setup procedure:

Setting up the DXP for operation with a new preamplifier is not difficult if carried out in a methodical manner. Here we sketch the procedure as a road map to the sections that follow. In the following Section 5 we will present procedures for selecting parameters for operation with a given x-ray spectrum.

Matching the DXP to a new preamplifier:

Preliminary preamplifier measurements: these measurements determine the polarity of the preamplifier, its gain, the range of its reset ramp (assuming a reset type preamplifier) , the risetime of its signals and the time it takes to reset.

4.2. Preliminary Preamplifier Measurements:

BEFORE CONNECTING THE DXP TO THE PREAMPLIFIER you should measure the following preamplifier parameters: 1) signal polarity; 2) reset range; 3) gain; 4) 10-90% pulse risetimes; and 5)reset time.

These quantities may be measured by the following steps:

1) Connect the preamplifier output to the input of a high impedance (1 MΩ) fast (100 MHz BW or greater) oscilloscope, preferably a digital oscilloscope, and place a radioactive source (typically ^{55}Fe) in front of the detector at a distance estimated to give a modest count rate (a few thousand cps).

2) Set the voltage range to 1-2 V/division and adjust the time base to observe the preamplifier's full signal range through several resets, as shown in the example of Figure 4.1.

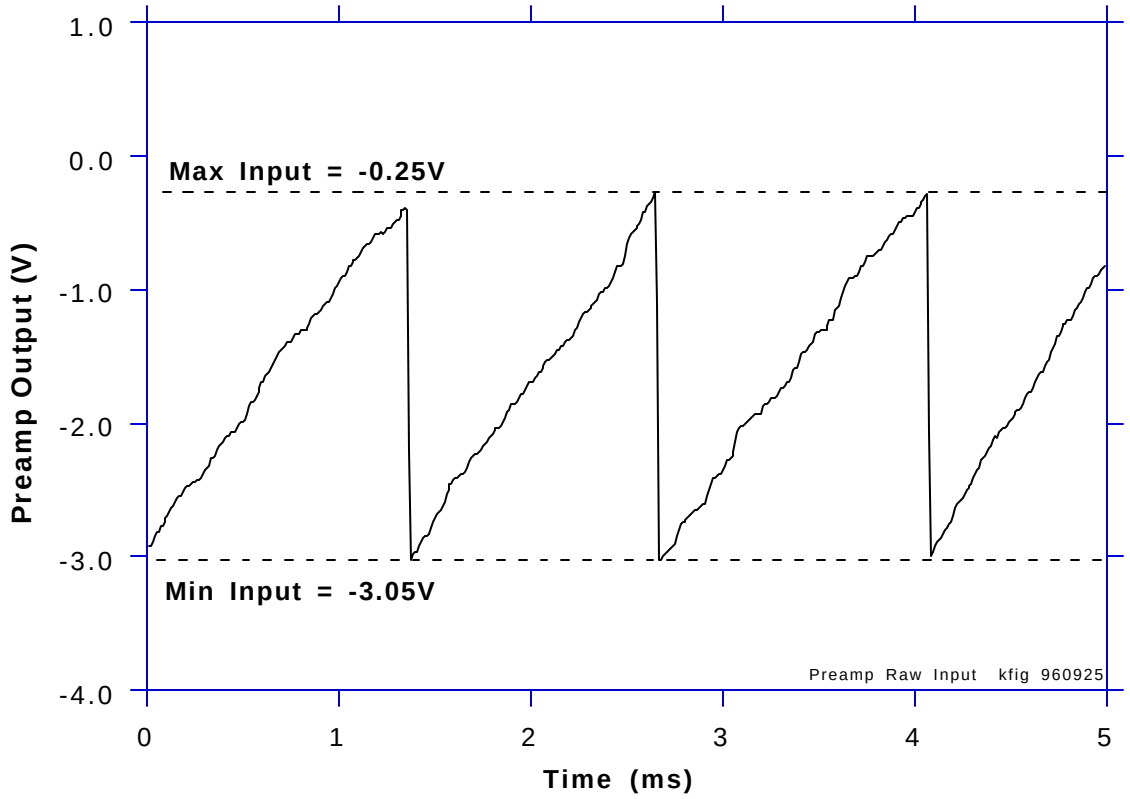


Figure 4.1: Preamplifier output waveform measured with a 10 Meg ohm scope probe. This detector has positive polarity x-ray signals and ramps over a 2.8 volt range between resets.

3) Record the signal polarity. "Positive" means the incoming x-rays cause the signal voltage to ramp upwards, as in Fig. 4.1, "negative" means they cause it to ramp downwards.

4) Record the maximum, minimum, and range values of the reset ramp (e.g. V_{max} = -0.25 V, V_{min} = -3.05 V, ΔV_{range} = 2.80 V in the example of Fig. 4.1). Also compute and record V_{mean}, the *mean ramp voltage* (e.g. -1.65 V in the example of Fig. 4.1).

5) Record the preamplifier's *reset voltage* V_{reset} (i.e. the voltage just after reset).
For positive polarity preamplifiers: $V_{reset} = V_{min}$ (as in Fig. 4.1).
For negative polarity preamplifiers: $V_{reset} = V_{max}$.

6) If necessary, offset the signal so that some part of the repeating ramp passes through 0 volts. Change the vertical gain to 2 - 10 mV/division (depending on the preamplifier) and decrease the time base (toward 1 µs/division) until individual x-ray events can be observed. This process is simplified it the scope trigger can be set for 0 volts with the same polarity as the amplifier so that the x-ray pulses trigger the scope. With a digital scope, the best course of action is to capture single event traces for study, as shown in Fig. 4.2.

7) Compute and record the preamplifier's gain G_p in mV/keV. This is found by dividing the amplitude of the captured x-ray pulse (in mV) by its known energy (in keV) from the radioactive source. ^{55}Fe, for example, produces Mn K_α x-rays at 5.9 keV. In the example of Fig. 4.2, the average step height is 21.5 mV, for a gain of 3.64 mV/keV.

8) Further reduce the time base until the rise time of individual x-ray events can be observed, as in the example of Figure 4.3.

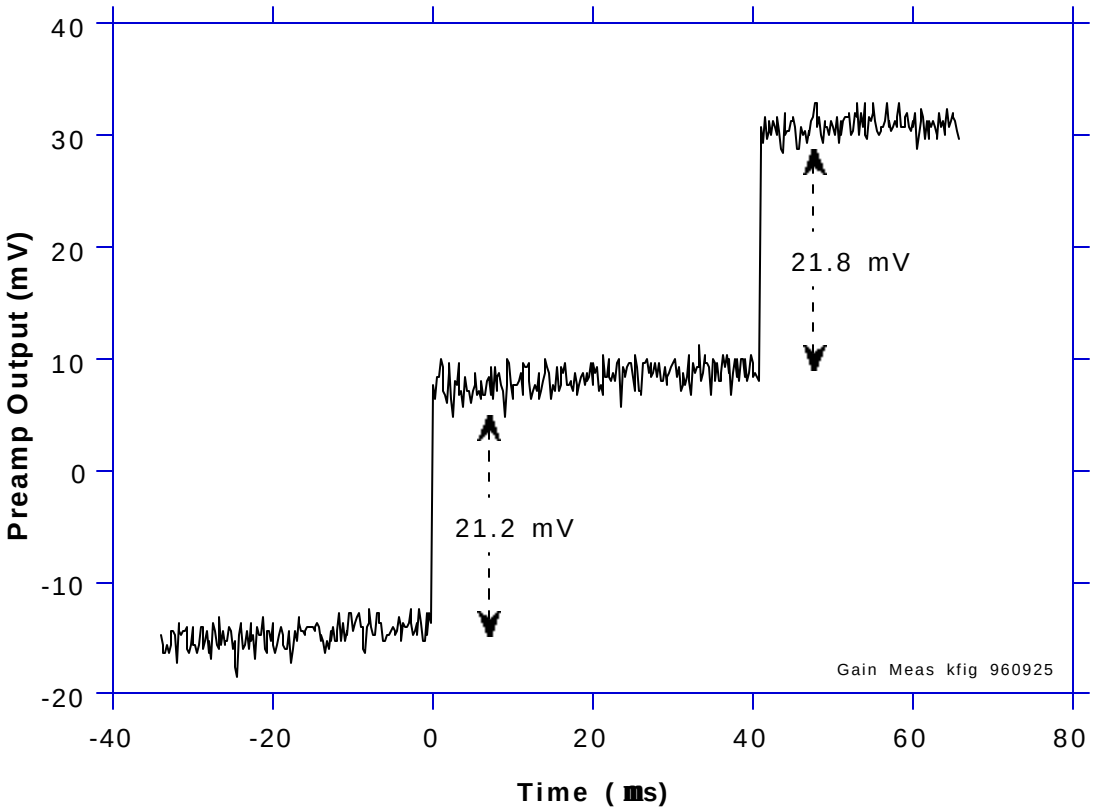


Figure 4.2: Mn K_α x-ray pulses recorded using a digital oscilloscope. The average step height is 21.5 mV, for a gain of 3.64 mV/keV.

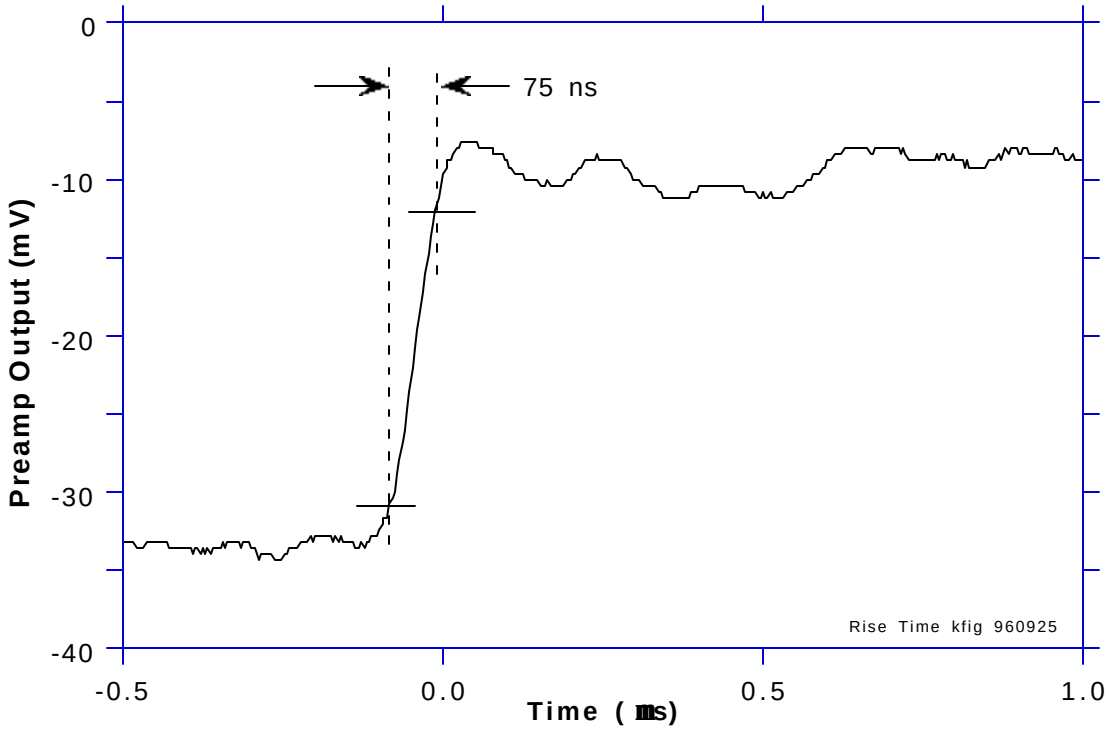


Figure 4.3: Mn K_α x-ray pulse recorded using a digital oscilloscope with a time base of 200 ns per division. The 10-90% risetime is measured at 75 ns.

9) Capture an event and measure and record its 10-90% risetime. This time will be useful in setting the gap periods of the DXP's trapezoidal filters. Note that a relatively short risetime is required if the detector is going to achieve very high counting rates with good pileup rejection. It is not particularly effective, for example, to attempt to use a peaking time of 0.5 μs with a preamplifier which has a 600 ns risetime. A well designed and constructed modern semi-conductor detector, attached to a properly matched preamplifier, should easily be able to achieve risetimes in the 200 to 100 ns range.

10) Measure the time it takes for the preamplifier to reset. Set the trigger opposite the polarity of the pulses (ie, for a negative polarity detector, trigger on the rising edge), which should trigger on the reset. Adjust the oscilloscope gain and time base so that the entire reset can be seen. Measure the time between the beginning of the reset and when the normal ramp returns. Typically, this time is 10 microseconds or less.

11) If the detector is a multi-element array, repeat these measurements for all elements. Provided the results are recorded carefully, these measurements will not have to be repeated unless the preamplifiers' gains are changed.

5. DXP Module Setup and Use: Overview:

To control the DXP module and use it to take data, host computer software is required to send commands to the DXP and transfer data to and from its various registers and memory locations.

PLEASE NOTE: THE FOLLOWING DISCUSSION THEREFORE ASSUMES THE EXISTENCE OF FUNCTIONING HOST COMPUTER CONTROL SOFTWARE CAPABLE OF DOWNLOADING PARAMETERS TO THE DXP AND EXERCISING ITS SOFTWARE ROUTINES. You can obtain such software from various commercial or National Laboratory sources, write your own (using primitives and midlevel libraries supplied by XIA: see the Host Software Manual for further information), or use LabView programs from XIA. If you need advice on this topic, call XIA or visit XIA's web site (http://www.xia.com/) for information, application notes, and lists of currently available programs.

5.1. Jumper Settings on the DXP-2X

There are several jumpers which affect the operation of the DXP-2X. These jumpers can be separated into two classes: system jumpers and channel jumpers. In general, the jumpers are set at the factory to be correct for almost all applications; however, some users may need to change the jumper settings for their particular application.

The system level jumpers determine the logic level used for the front panel digital signals on the DXP-2X (on the Model C, there is only one input – GATE, while the Model T has three – GATE, SYNC and AUX). The GATE and SYNC signals are both inputs to the DXP, and can be configured to use either TTL or NIM logic levels. The AUX signal can be an input or an output if configured to use TTL levels, but can only be used as a NIM output. By default, the jumpers are set so that TTL logic levels are selected. The system jumpers are listed in **Table 5.1** below, and are located just behind the LEMO connectors on the front panel. The jumpers have three pins; connecting the two pins closest to the front panel selects TTL levels, while connecting the two pins furthest from the front panel selects NIM levels (the positions are labeled on the board). Note that for the SYNC and GATE logic levels, the corresponding jumpers must be set in pairs. The logic level indicator jumper allows the host to determine the jumper settings, as long as all the jumpers are set to the same position. A mixed logic level system is permissible, although in this case the indicator jumper will have little meaning.

Table 5.1: System Jumpers

Jumper	Function
JP6, JP7	SYNC Logic level
JP8, JP9	GATE logic level
JP10	AUX logic level
JP11	Logic level indicator (mapped to the General Status Register)

There are three jumpers associated with each channel. These jumpers determine the gain of the input buffer, the ASC tracking mode (RAMP for reset preamplifiers or OFFSET for feedback preamplifiers), and whether the analog inputs are treated as single-ended (shield connected to GND on the board) or pseudo-differentially. The default jumper settings select a unity input buffer gain, ramp mode, and a single-ended input. A unity gain buffer at the input is more appropriate for almost all situations; the x¼ setting is useful for preamplifiers with a reset range extending beyond ±10 Volts, or for detectors with such extremely high gain that the extra attenuation is necessary to properly keep the signal in the ADC's range. The jumpers are arranged around the input relay for each channel, just behind the input SMA front panel connector. The channel jumpers are listed below in **Table 5.2.** All jumpers are labeled as indicated in the table.

Table 5.2: Channel Jumpers

Jumper	Function
JPx00, where x=1, 2, 3, 4 for DXP Channel 0, 1, 2, 3	Input buffer gain: x1 or x¼
JPx01	Input type: single-ended (GND) or pseudo-differential (DIFF)
JPx04	Tracking mode: RAMP or OFFSET

5.2. Overview to DSP Configuration, Parameter Download and Run:

At the highest level, the following actions are required to start up the DXP and use it to collect data:

1) Initialize FiPPI and DSP configurations: After the DXP module is powered up, the DSP program and FiPPI configuration need to be downloaded. Each is read from an ASCII file and sent to the DSP via single word or block data transfers. This is described further in Appendix C.

2) Download data acquisition and control parameters: As noted in §3, the user can set a number of parameters which control the acquisition of data, including the FiPPI filter lengths, thresholds and other options. These parameters reside in the DSP's internal data memory, and are described in §3 and in detail in the DSP Software Manual [Ref. 1]. For clarity in the text, their names are shown in all capital letters. Also as described in the §3 overview, all parameters can (and should) be accessed symbolically in order to assure host software compatibility across DSP code versions. In this method, the host machine uses a lookup table supplied with the DSP code to translate each parameter name (such as SLOWLEN, the slow filter length) into a data memory address in the DSP. To write the parameter value, the host machine then sets the channel number in the GCR (or sets the broadcast bit to write to all channels), writes the parameter's address to the Transfer Start Address Register (TSAR)for the specified channel(s), and then writes the data to the DSP('s) with a single word or block transfer. DSP data transfers are described more fully in Appendix B.

3) Start data taking: This is described more fully in Section 8. Normal data acquisition is initiated by writing the RUNTASKS parameter with its "special_run" bit cleared, and then writing to the GCR with the "RunEna" bit set. Before taking data, the DSP ensures that the FiPPI parameters are properly downloaded and the front end parameters are set properly; calibration procedures are performed if necessary (if the gain changed, for example).

4) Stop data taking: Data acquisition is terminated by writing to the GCR with the "Run_Ena" bit cleared.

5) Upload acquired data and statistics: The MCA data are typically read out by a block mode read of the MCA spectrum, located in program memory. The offset of the spectrum within program memory is stored in the parameter SPECTSTART. The host machine sets the channel bits in the GCR, writes the address of the spectrum to the TSAR, and then reads the data from the DSP with a single word or block transfer. DSP data transfers are described more fully in Appendix B.

Steps 1 - 2 need only be carried out once when the DSP is initially powered up or if the program is reloaded. Steps 3 - 5 are carried out once per set of data to be collected. The DSP code is described in more detail in the following section. Then, in Section 7, we will discuss the data collection process in more detail.

6. DXP-2X DSP Code Description

6.1 Introduction and Program Overview

The following sections are intended to provide the DXP user with a good understanding of the various tasks performed by the DSP in each channel of the DXP-2X. The DSP performs several functions:

1) Respond to input and output calls from the host computer to start and stop data collection runs, download control parameters, and upload collected data.
2) Perform system calibration measurements by varying the various DAC settings under its control and noting the output change at the ADC.
3) Make initial measurements of the slow filter baseline and preamplifier slope value at the start of data taking runs to assure optimum starting parameter values.
4) Collect data:
   a) Read energy values E_x from the FiPPI, under interrupt control, and store them in DSP buffer memory in less than 0.25 µs.
   b) Adjust the ASC control parameters, under interrupt control, to maintain its output within the ADC's input range.
   c) Process captured E_x values to build the x-ray spectrum in DSP memory.
   d) Sample the FiPPI slow filter baseline and build a spectrum of its values in order to compute the baseline offset for E_x values.

Several DSP program variants are available to cover a range of applications. The standard program (Variant 01) provided with the DXP-2X is for typical x-ray fluorescence spectroscopy using a pulsed reset preamplifier. Additional program variants, listed in Table 6.1, are available for other applications, including hardware diagnostics. Other specialized measurements, including: 1) x-ray mapping; 2) Quick XAFS scanning; 3) switching between multiple spectra synchronously with an experimentally derived signal (e.g. "Phased locked EXAFS"); and 4) time resolved spectroscopy (e.g. "multi-channel scaling"). The variants which have been released to date are described in Section 5.11. Several other variants have been developed for particular customers and may be made available upon request.

Table 6.1: DSP Software Variants

Variant	Name	Standard Application/Configuration
0	D2XR01	General x-ray spectroscopy data acquisition for pulsed reset preamplifiers Single MCA per channel with up to 8K bins
1	D2XF01	General x-ray spectroscopy data acquisition for RC feedback preamplifiers Single MCA per channel with up to 8K bins

By convention, the DSP programs are named "D2XPmmnn.HEX", where P is the preamplifier type, mm is the code variant and nn is the version numbers. The hex file format is in ASCII, with the parameter table at the top followed by the code generated by the Analog Devices 218x development system.

The internal data memory area is subdivided into three sections. The first section, starting at location 0x4000, contains DSP parameters and constants, both those used for controlling the DSP's actions and those produced by the DSP during normal running. These parameters and their addresses are listed and described in the following sections. When these parameters are referred to they will be denoted by all capital letters (e.g. RUNTASKS). The locations of parameters can (and, for forward compatibility must) be determined from the symbol table (ie, lookup by name).

The second section of data memory contains acquired monitoring data such as the baseline event histogram. The third section of internal data memory is used as a circular buffer for storing events from the FiPPI. Note that future hardware revisions may eliminate the need for this buffer area, in which case it could be switched to more histogramming area.

6.2.Program Flow

The flow of the DSP program is illustrated in Figure 6.1. It is essentially identical for all program variants. The structure is very simple; after initialization, the DSP enters an idle phase, waiting for a signal from the host to start a run. During this idle phase, the DSP is continuously collecting baseline events from the FiPPI as well as monitoring the Analog Signal Conditioner (ASC) to keep the ADC input signal in the proper range and to adjust the slope generator to match the current input rate. When the Begin Run signal is received (from the host through the GCR register), the DSP first determines whether the run is a normal data-taking run or a special run.

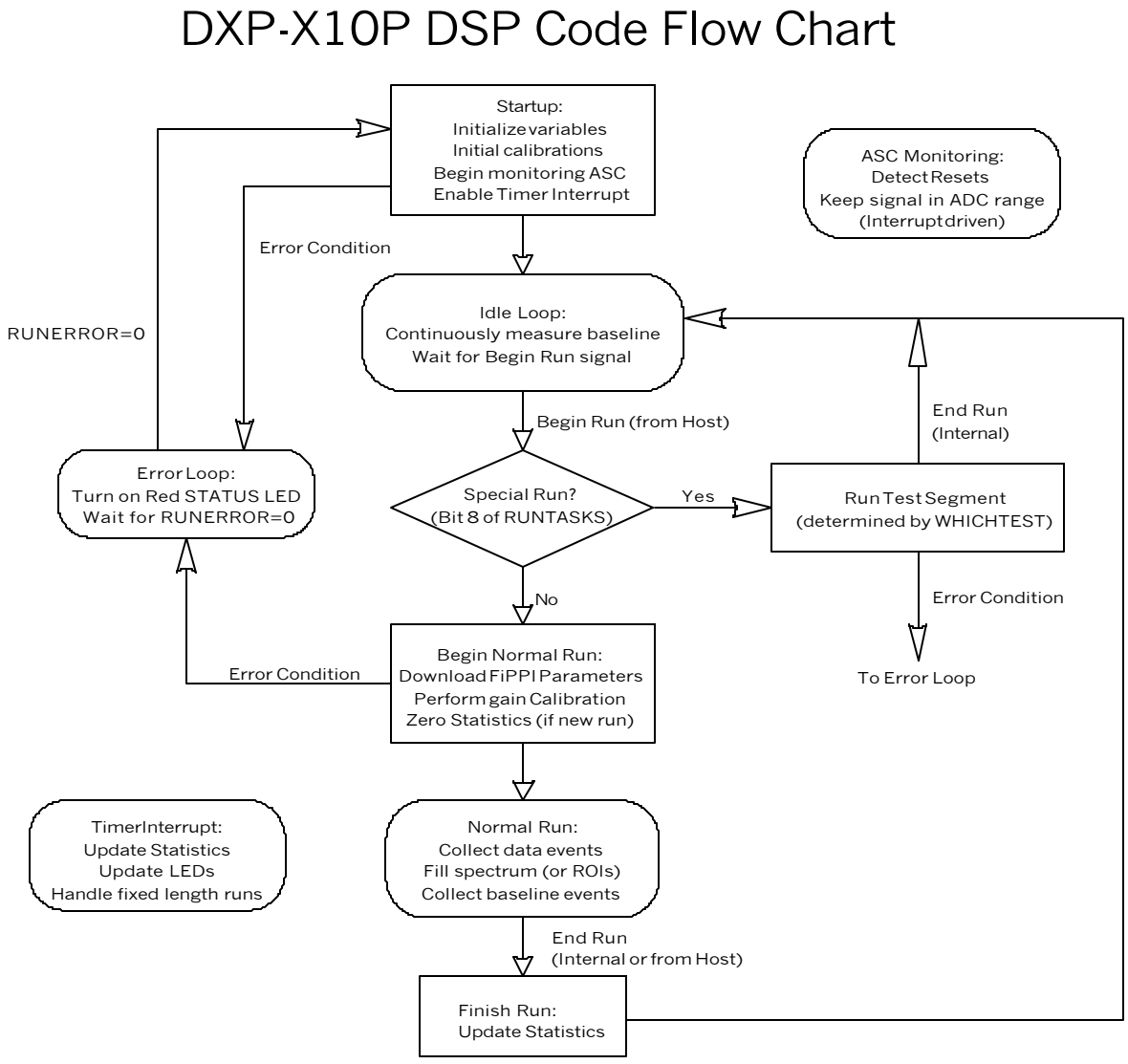


Figure 6.1: DSP code flow diagram

In a normal run, ASC monitoring and baseline collection continue as in the idle phase. Event interrupts are enabled; when the FiPPI detects an event, it interrupts the DSP, which quickly responds and reads the energy value from the FiPPI into an internal buffer in data memory. The events in the buffer are then used to build the x-ray spectrum (or fill regions of interest).

In a special run, the action is determined by the value of the parameter WHICHTEST. The special runs include calibration tasks such as collecting an ADC trace, as well as ways of putting the DSP code into a special state (such as putting it into a dormant state to allow reprogramming the FiPPI on the fly). Normally, the special runs end on their own and the DSP returns to the idle state.

After the initialization phase, the Timer interrupt is enabled. This interrupt is used to handle the housekeeping type chores, such as updating the statistics during a run, controlling the rate LED, and handling fixed length runs. The Timer interrupt occurs with a period of 500 µsec.

If the DSP encounters an error condition, the DSP turns on the red status LED and waits for the host to set the parameter RUNERROR to 0 (after finding and fixing the problem that resulted in the error condition).

Each phase of the DSP program is discussed in more detail below.

6.3. Initialization

The DSP code starts running immediately after the DSP download is complete. During the initialization phase, several tasks are performed:

1) Setup internal DSP control registers
2) Zero spectrum and data memory, then initialize parameters to default values.
3) Set ASC DACs to initial default values
4) Initialize FiPPI and download default filter parameters
5) Perform initial calibrations for controlling the ASC:
   a) Find the SlopeDAC setting corresponding to zero slope
   b) TrackDAC Calibration (determine TrackDAC step needed to move the ADC input signal from the edge of the range to the center of the range)
   c) Measure conversion factor used to calculate the contribution of the slope generator to the FiPPI baseline.
6) Enable the input relay and enable the ASC and timer interrupts.

After the interrupts are enabled, the DSP is alive and ready to take data. After completing the initialization phase, the DSP enters the idle phase. In the idle phase, the DSP continuously samples the FiPPI baseline and updates the baseline subtraction register in the FiPPI so that the FiPPI is always ready to take data.

6.4. Event Processing

There are two primary tasks performed during a normal data-taking run: event processing and baseline processing. These tasks are described in detail below.

6.4.1. Run Start

Prior to the start of a normal the run, the DSP performs several tasks:

- Sets the desired gain. If the gain has changed, the TrackDAC calibration is redone (for reset detectors only).
- Sets the desired polarity (the internal DSP polarity and the FiPPI polarity must be changed simultaneously to avoid ASC instability). Only applicable if the desired polarity differs from the default negative polarity (and then only for the first run).

- Downloads the specified FiPPI parameters (SLOWLEN, SLOWGAP, etc) to obtain the desired peaking time.
- Updates the internal calibrations with the new gain and FiPPI values.
- If desired, the run statistics and the MCA are cleared (determined by the ResetMCA bit in the GCR). Otherwise, the run is treated as a continuation of the previous run. Note that for a run continuation, no gain or FiPPI changes are performed. In either case, the run number (parameter RUNIDENT) is incremented.

6.4.2 Event Interrupt

When the FiPPI detects a good event, it triggers a high priority interrupt in the DSP. Upon receiving the interrupt, the DSP immediately reads the event energy from the FiPPI into an internal circular buffer and increments the write pointer into that buffer. The normal event loop compares the write pointer to the read pointer to determine that there is a new event to process.

6.4.3 Event Loop

The processing that takes place during a normal collection run is very simple, in order to allow high event rates. The structure of the event loop is illustrated below in pseudocode:

```
while (RunInProgress)
        {
        if (EventToProcess)
                ProcessEvent
        else
                CollectBaseline
        endif
        }
RunFinish
goto IdleLoop
```

The run can be stopped by the host by clearing the RunEnable bit in the CSR, or can be stopped internally for fixed length runs; see Section 6.10.5 below.

The event processing involves either binning the energy into an MCA or determining whether the event falls into a defined SCA window, depending upon the DSP code variant.

If there is no event to process, the DSP reads a baseline value from the FiPPI; see below for a detailed description of the baseline processing.

Once the run is over, the statistics are finalized and the DSP returns to the idle state where it continuously samples baseline and waits for a command to start a new run.

6.4.4 Spectrum Binning

The primary event processing task is to use the energies measured in the FiPPI to build up a full energy spectrum (MCA). The MCA bin width is determined by the analog gain, the FiPPI filter length, and the binning parameter BINFACT1. The DSP determines the spectrum bin by multiplying the FiPPI energy output by (1/BINFACT1). If the bin is outside the range determined by the parameters MCALIMLO and MCALIMHI, the event is classified as an underflow or overflow. Otherwise, the appropriate bin is incremented. A 24-bit word is used to store the contents of each bin, allowing nearly 16.8 million events per MCA channel.

6.4.5 SCA Mapping

An alternate variant of the DSP code allows the user to define up to 16 or more SCA regions and count the number of events that fall into each region. The regions are defined in terms of MCA bin number, and can overlap. A useful method for defining the SCA windows is to take a run with the full MCA spectrum, and use the spectrum as an aid in choosing the limits for each SCA. The reduced amount of data storage in SCA mapping mode is very useful in time resolved spectroscopy or scanning applications, where separate spectral data are desired for many different time or spatial points.

6.5 Baseline Measurement

The DSP collects baseline data from the FiPPI whenever there are no events to process, both during a run and between runs (when there are never events to process). The DSP keeps a running average of the most recent baseline samples; this average is written back into the FiPPI where it is subtracted from the raw energy filter value to get the true energy. The baseline data read from the FiPPI is just the raw output of the energy filter. One bit of the baseline register is used to indicate whether the sample occurred while an event was in progress, in which case it is not used.

Two methods are available to determine the average baseline value. By default, an *infinite impulse response (IIR)* filter is used, where the baseline average is calculated by combining a new baseline sample with the old average, using weights x and (1-x) respectively, where x is typically 1/64. By setting the appropriate bit in the parameter RUNTASKS (see below), a *finite impulse response (FIR)* filter is used, where the baseline mean is just the straight average of the N most recent baseline samples. Both averaging methods are described in more detail in the following sections. The baseline mean is stored with 32 bit precision in the parameters BASEMEAN0 (high order word) and BASEMEAN1.

6.5.2 IIR (Infinite Impulse Response) Filter

By default, the baseline mean is calculated using an infinite impulse response filter, characterized in the following way:

$$< B_i >= \frac{N-1}{N} < B_{i-1} > + \frac{1}{N} B_i$$

where <*Bi*> is the baseline mean after the *i*th baseline sample, *Bi* is the *i*th baseline sample, and <*Bi-1*> is the baseline mean before the *i*th sample. With this filter, the most recent baseline samples are weighted the most, but (up to the precision of the stored mean value) all baseline values have a small effect on the mean (hence the infinite in the name).

The length of the filter is controlled by the parameter BLFILTER, which holds the value 1/N in 16 bit fixed point notation, which has 1 sign bit and 15 binary bits to the right of the decimal point. Expressed as a positive integer, BLFILTER = (1/N)*2**15. The default value for BLFILTER corresponds to N=64. Interpreting BLFILTER as an integer gives (1/64)*2**15 = 2**9 = 512.

6.5.3 FIR (Finite Impulse Response) Filter

By setting the appropriate RUNTASKS bit, it is possible to choose a finite impulse filter to calculate the baseline mean. With this filter, a straight average of the N most recent valid baseline samples is used to calculate the mean. To implement this filter, a buffer large enough to hold all N samples is necessary. For this reason, the length of the finite response filter is limited to 1024. The filter length is stored in the parameter BLFILTERF.

6.5.4 Baseline Histogram

As part of the baseline processing, all valid baseline samples are entered into the baseline histogram, which occupies 1024 words of data memory. The baseline histogram can be very useful in monitoring or

evaluating the performance of the DXP-X10P. The parameter BASESTART contains the pointer to the location of the histogram in data memory, and the length (nominally 1K) is contained in the parameter BASELEN.

The baseline histogram is centered about a zero baseline. The parameter BASEBINNING determines the granularity of the histogram; 2**BASEBINNING baseline values are combined into one bin of the baseline histogram. The default value of BASEBINNING is 2 (i.e., the baseline value is divided by 4 to determine the bin). All valid baseline values are included in the histogram, even if there is a baseline cut in use.

The baseline histogram is only filled during a normal datataking run; when the DSP is idle, the baseline average is calculated but the histogram is not filled. Since the baseline histogram is stored in data memory, 16-bit words are used to record the bin contents. As a result, the histogram overflows quite often; the time to overflow depends on the baseline sample rate (typically several 100 kHz) and the width of the baseline distribution. When the DSP detects an overflow, all bins are scaled down by a factor of 2 and histogramming continues.

The baseline distribution should be very gaussian; the width of the distribution reflects the electronic noise in the system (including the effects of the energy filter). A tail on the positive side of the distribution indicates the presence of energy in the baseline, resulting from undetected pileup or energy depositions that did not satisfy the trigger threshold. The tail should be very small compared to the peak of the histogram; it will grow with rate. If this tail is too large, it can have a noticeable effect on the baseline mean, leading to negative peak shifts. Under these circumstances, enabling the baseline cut is useful in eliminating the bias.

A tail on the low energy side of the baseline distribution is usually caused by baseline samples just after a preamplifier reset; the effects of the reset can last quite a while (tens of microseconds), especially for optical reset preamplifiers. It is usually best not to take data while the reset is in effect; the dead time associated with a reset can be adjusted using the parameter RESETWAIT, which sets the dead time in units of 250 ns.

6.5.5 Residual Baseline

When operating with a reset type preamplifier, the raw baseline measured in the FiPPI (which is just the output of the energy filter) comes from two sources: the detector preamplifier and the slope generator in the DXP-X10P itself. At high rates, the slope gets rather large in order to balance the high energy deposition rate in the detector; under these conditions, the baseline due to the slope is by far the dominant factor in the baseline.

By default, the DSP continually adjusts the slope to match the current rate; these slope adjustments result in an instantaneous change in the baseline. If the baseline due to the slope generator is included in the baseline mean, the change in the calculated mean would be delayed relative to the change in the slope, due to the effect of all the baseline samples prior to the slope change. For this reason, the baseline due to the slope is subtracted out of the overall baseline prior to calculating the mean value (and added back in prior to loading the FiPPI baseline subtraction register). The *residual* baseline included in the mean reflects the detector leakage current, and should be fairly constant with rate (to the extent that the leakage current does not depend on rate). The calibration procedure used to determine the baseline due to the slope generator is performed during the initial startup procedure.

By default, the baseline due to the slope generator is taken out of the baseline average. The user can choose to include the slope baseline in the mean by clearing the residual baseline bit (6) in RUNTASKS.

6.5.6 Baseline Cut

As specified above, a baseline cut is available to exclude baseline samples that include real event energy, which can lead to peak shifting at high event rates. The cut is expressed as a fraction of the peak value of the baseline distribution; by default, the baseline cut is set to 5%. The cut values are based on the baseline histogram, and are recalculated every time the histogram overflows (every few seconds). The DSP searches on either side of the peak of the baseline distribution for the first bin whose contents are less than the cut (.05 by default) times the peak value; these bin numbers are used to calculate the actual baseline cut.

The cut fraction is stored in the parameter BLCUT, expressed in 16-bit fixed-point notation. Interpreted as an integer, BLCUT = (cut fraction)*2**15; the default 5% cut corresponds to BLCUT=1638 decimal (or 666 hex). The actual cut values determined by the DSP code are stored in BLMIN and BLMAX. The baseline cut is enabled or disabled by setting or clearing a bit (10) in the RUNTASKS parameter.

6.6 Interrupt Routines

There are several tasks performed under interrupt control within the DSP on the DXP-X10P. The event interrupt routine (which just transfers event data from the FiPPI to an internal buffer) is described above in Section 0 above. There are two other interrupt routines: the ASC interrupt is used to keep the analog signal within the input range of the ADC, and the timer interrupt is used to handle such housekeeping chores as updating statistics. These routines are described in more detail below.

6.6.2 ASC Monitoring

There are four main tasks performed by the ASC interrupt routine:

- Detects Resets (reset detectors only)
- Adjusts the slope generator to match the event rate (reset detectors only)
- Adjusts the offset value to keep the signal in range (feedback detectors only)
- Moves the signal back to the center of the ADC range whenever it drifts out of range (high or low)

The ASC interrupt routine is triggered whenever the FiPPI detects the ADC going out of range. If the out of range is due to the signal drifting out of range (instead of a reset), the DSP triggers a TrackDAC step to bring the signal back to the center of the ADC range, and data taking resumes. If the DSP determines that the out of range is due to a reset, then the DSP holds the signal at the center of the ADC range for a time determined by the parameter RESETINT, which specifies the dead time after a reset in 0.25 μsec units. After the reset interval, the signal is released and data taking resumes.

The DSP keeps track of how many times the signal drifts out of range in both directions, and adjusts the slope such that the number of drifts high (DriftUps) roughly matches the number of drifts low (DriftDowns). If the DSP determines that the slope must be changed to match the rate, the SlopeDAC value is modified by a constant fraction of the parameter SLOPEVAL determined by the value of the parameter SGRANULAR. By default, the slope adjustment granularity is 5%, which is a good compromise between adjusting the slope quickly to match quickly changing input rates and being able to set the SlopeDAC just right.

For a feedback detector, the offset added to the input signal is adjusted such that the signal stays in range as much as possible.

6.6.3 Timer Interrupt

Every 500 μsec, the DSP is interrupted to take care of the regular 'maintenance' type tasks. These tasks include:

- Update the run statistics LIVETIME, REALTIME and FASTPEAKS (only during a run).
- Control the Rate LED. This LED flashes whenever a reset is detected (reset detector only), and during a run the color indicates the current output/input ratio. By default, the LED flashes green for OCR/ICR>0.5, flashes yellow (green plus red) for 0.5>OCR/ICR>1/e, and flashes red for OCR/ICR<1/e. The thresholds are determined by the parameters YELTHR and REDTHR.
- Handle fixed length runs. During a fixed length run, the current value of EVTSINRUN (output events), FASTPEAKS (input events), LIVETIME or REALTIME is compared to the desired run length. Once the value exceeds the desired value, the run is ended.

6.7 Error Handling

When the DSP detects an error in the operation of the DXP-X10P, the red Status LED is turned on, and the source of the error is stored in the parameter RUNERROR. The possible values for RUNERROR are listed below:

RUNERROR Value	Meaning
0	No Error
1	FiPPI communication error
2	ASC setup failure
3-5	Reserved
6	TrackDAC calibration error

A FiPPI communication error could mean that the FiPPI configuration was not successful. An ASC calibration error can indicate a hardware problem, or possibly that a jumper is not set properly (for example, the DSP code for reset preamplifiers will generate an error if the jumper is set to run in OFFSET mode).

Once the source of the error has been located and cleared, the host can set RUNERROR to 0 to force the DSP to exit the error loop and reinitialize the system. Note that all system settings are saved when initialization is performed coming out of the error loop. Of course, another valid method for clearing the error is to redownload the DSP code after fixing the problem.

6.8 Specifying Data Acquisition Tasks (RUNTASKS):

Many aspects of the operation of the DXP-2X are controlled by bits in the RUNTASKS parameter. The meaning of each RUNTASKS bit is described below:

Table 6.2: Description of RUNTASK parameter control bits

Bit	Meaning if set (1)	Meaning if cleared (0)
0	Reserved (set to 0)	Reserved (set to 0)
1	Update SlopeDAC or OffsetDAC value to match current rate (DEFAULT)	SlopeDAC or OffsetDAC adjustments disabled
2	Use Finite Impulse Response (FIR) filter to calculate baseline average	Use Infinite Impulse Response (IIR) filter to calculate baseline average (DEFAULT)
3	Acquire baseline values for histogramming and averaging (DEFAULT)	Disable baseline acquisition
4	Adjust fast filter threshold to compensate for rate shifts	Disable fast filter threshold adjustment
5	Correct for baseline shift, either in FiPPI (reset) or DSP (feedback) (DEFAULT)	Disable baseline correction
6	Apply residual baseline correction (DEFAULT)	No residual baseline correction
7	Continuously write baseline values to baseline history circular buffer (DEFAULT)	Disable writing baseline values to baseline history circular buffer (useful when reading the buffer)
8	Indicates special task or calibration run specified by WHICHTEST	Indicates normal acquisition run (DEFAULT)
9	Histogram DeltaBaseline (baseline - <baseline>)	Histogram raw baseline (DEFAULT)
10	Enable baseline cut	Disable baseline cut (DEFAULT)
11-15	Reserved (set to 0)	Reserved (set to 0)

6.9 Special Tasks (WHICHTEST)

Special tasks are selected by starting a run with bit 8 of the RUNTASKS parameter set. The following tasks are currently supported:

Table 6.3: Special tasks and test segments which can be selected with the WHICHTEST parameter

Number	Test Segment
0	Set ASC DAC values to current value of GAINDAC, SLOPEDAC and/or OFFSETDAC
1	Acquire ADC trace in history buffer
2	Gain calib (measure TDACPERADC)
3	Slope calibration (measure SLOPEMULT)
4	Measure ADC non-linearity
5	Not currently used
6	Put DSP to sleep while FPGA logic is downloaded
7	RESET calibration (measure TRACKRST)
8	OffsetDAC calibration (measure OFFDACVAL)
9-10	Not currently used
11	Program Fippi
12	Set internal polarity to current value of POLARITY parameter
13	Close input relay
14	Open input relay
15	RC feedback calibration trace of baseline filter and decimator values
16	RC feedback calibration trace of event filter and decimator values

6.10 DSP Parameter Descriptions

As noted above, DSP operation is based on a number of parameters. Some are control parameters required to operate the DXP, some are calibration values determined by the DSP, and others are run statistics.

Table 6.4: Summary of DSP parameter definitions

Variable	Type	Description	Reference
PROGNUM	Constant	Program variant number.	0
CODEREV	Constant	Current DSP program revision.	0
HDWRVAR	Constant	Hardware variant. DSP reads this from interface FPGA.	
FIPPIREV	Constant	FiPPI design revision. DSP reads this from FiPPI FPGA.	
FIPPIVAR	Constant	FiPPI design variant. DSP reads this from FiPPI FPGA.	
DECIMATION	Constant	Slow filter decimation factor. DSP reads this from FiPPI FPGA.	
RUNIDENT	Returned	Run identifier	
RUNERROR	Returned	Error code if run is aborted, 0 for success	6.7
BUSY	Returned	DSPs current acquisition status. Values listed below.	
Acquisition Statistics:			
LIVETIME0,1,2	Statistic	DAQ live time in 800 nsec units	
REALTIME0,1,2	Statistic	Elapsed acquisition time in 800 nsec units	
EVTSINRUN0,1	Statistic	Number of events in MCA spectrum	
UNDRFLOWS0,1	Statistic	Number of MCA underflow events	
OVERFLOWS0,1	Statistic	Number of MCA overflow events	
FASTPEAKS0,1	Statistic	Number of input events detected by FiPPI	
NUMASCINT0,1	Statistic	Number of ASC interrupts	
NUMRESETS0,1	Statistic	Number of "reset" events seen	
NUMUPSETS0,1	Statistic	Number of "upset" events seen	
NUMDRUPS0,1	Statistic	Number of "drift up" events seen	
NUMDRDOS0,1	Statistic	Number of "drift down" events seen.	
NUMZIGZAG0,1	Statistic	Number of "zigzag" events seen	
BASEEVTS0,1	Statistic	Number of baseline events acquired	
BASEMEAN0,1	Statistic	Updating mean baseline value	
Control parameters:			
WHICHTEST	Parameter	Which test segment to execute.	
RUNTASKS	Parameter	Which tasks will be executed in run sequence	6.8
BINFACT1	Parameter	MCA binning factor	
MCALIMLO	Parameter	Lower limit of MCA spectrum	
MCALIMHI	Parameter	Upper limit of MCA spectrum	
TRACEWAIT	Parameter	ADC trace time factor	
ASCTIMOUT	Parameter	Timeout for ASCSetup in tenths of seconds	
YELLOWTHR	Parameter	Medium rate throughput threshold for front panel LED	
REDTHR	Parameter	High rate throughput threshold for front panel LED	
PRESET	Parameter	Preset type (0:none; 1:real time; 2:live time; 3: output cts; 4: input cts)	0
PRESETLEN0,1	Parameter	Preset run length	0
FiPPI Digital Filter/Event selection parameters:			
SLOWLEN	Parameter	Slow filter length	
SLOWGAP	Parameter	Slow filter gap	
PEAKINT	Parameter	Peak interval	
FASTLEN	Parameter	Fast filter length	
FASTGAP	Parameter	Fast filter gap	

THRESHOLD	Parameter	Threshold value for fast filter trigger (range: 1-255, 0 disables)	
MINWIDTH	Parameter	Minimum peak width	
MAXWIDTH	Parameter	Maximum peak width	
SLOWTHRESH	Parameter	Threshold for slow filter trigger (range 1-255, 0 disables)	
PEAKSAM	Parameter	Peak sampling time	
Baseline Related Parameters:			
BLFILTER	Parameter	Filtering parameter for baseline (IIR filtering)	
BLFILTERF	Parameter	Filtering parameter for baseline (FIR filtering)	
BASEBINNING	Parameter	Baseline binning for histogram (0:finest to 6:coarsest)	
BLCUT	Parameter	DSP baseline cut (cut at BLCUT*FWHM, units defined below)	
BLMIN	Calibration	Min baseline value accepted in average (calculated from BLCUT)	
BLMAX	Calibration	Max baseline value accepted in average (calculated from BLCUT)	
ASC Control Parameters and Calibrations (all variants)			
POLARITY	Parameter	Preamplifier signal polarity (0:negative step; 1:positive step)	
GAINDAC	Parameter	Current Gain DAC value (16 bit serial DAC, range 0-65535).	
HIGHGAIN	Parameter	High gain relay setting	
INPUTENABLE	Parameter	Input Enable relay setting	
ASC Control Parameters and Calibrations (pulsed reset variants)			
RESETWAIT	Parameter	Quick Reset time, 25ns units	
RESETINT	Parameter	Reset time, 0.25 usec units	
SLOPEDAC	Calibration	Current Slope DAC value (16 bit serial DAC, range 0-65535)	
SLOPEZERO	Calibration	Slope DAC zero value (approximately center of range)	
SLOPEVAL	Calibration	Abs(SLOPEDAC-SLOPEZERO)	
SGRANULAR	Parameter	Slope DAC step size	
TRKDACVAL	Parameter	Tracking DAC value: 12-bit parallel	
TDACWIDTH	Parameter	Track DAC pulse width 50 ns units	
TDQPERADC	Calibration		
TDQPERADCE	Calibration		
ASC Control Parameters and Calibrations (RC feedback variants)			
OFFSETDAC	Parameter	Current offset DAC value (16 bit serial DAC, range 0-65535).	
OFFSETSTEP	Parameter	Offset DAC step size	
TAURC	Parameter	Preamplifer decay constant, in 25 ns units (RCF variant only)	
RCFCOR	Calibration	Preamplifer decay correction (RCF variant only)	
Miscellaneous Constants:			
SPECTSTART	Constant	Address of MCA spectrum in program memory	
SPECTLEN	Constant	Length of MCA spectrum buffer	
BASESTART	Constant	Address of baseline histogram in data memory (offset by 0x4000)	
BASELEN	Constant	Length of baseline histogram	
EVTBSTART	Constant	Address of event buffer in data memory (offset by 0x4000)	
EVTBLEN	Constant	Length of baseline histogram	
HSTSTART	Constant	Address of history buffer in data memory (offset by 0x4000)	
HSTLEN	Constant	Length of history buffer	
NUMSCA	Parameter	Number of SCA regions defined (mapping variants only)	6.4.5
SCAxLO, x=0-23	Parameter	Lower MCA channel for SCA region x (mapping variants only)	
SCAxHI, x=0-23	Parameter	Upper MCA channel for SCA region x (mapping variants only)	
USER1-USER8	User	User variables. Host software can use these for any purposes	

6.10.2 Firmware and Hardware Informational Parameters

6.10.3 Acquisition Statistics

6.10.4 Control parameters

6.10.5 Specifying fixed run lengths (PRESET,PRESETLEN0,1):

By default, the DXP-2X acquires data until the host clears the RunEna bit in the GCR.

A fixed run length can be specified using the parameters PRESET and PRESETLEN0,1, as follows:

PRESET specifies the type of run:

0 = indefinite (default)

1 = fixed realtime

2 = fixed livetime

3 = fixed output events

4 = fixed input counts

PRESETLEN0,PRESETLEN1 specifies the length of preset fixed length run, as a 32 bit quantity.

For fixed real time or live time, the units are 400 nanosecond intervals.

6.10.6 Setting the slow filter parameters (SLOWLEN,SLOWGAP)

In general the user does not modify these parameters directly, but through the host software routine SetAcquisitionValues (See Appendix).

The DXP uses a trapezoidal filter, characterized by the peaking time, T_p, and flat-top time, T_f. The peaking time is determined by the SLOWLEN and DECIMATION values. DECIMATION is automatically sensed by the DSP and should not be modified. For T_p and T_f. in µsec, the following gives the value of SLOWLEN and SLOWGAP:

$$SLOWLEN = 40*T_p *2^{-DECIMATION}$$

$$SLOWGAP = 40*T_f*2^{-DECIMATION}$$

The user will want to be able to choose the peaking time based on resolution and throughput requirements. Throughput (output count rate, OCR, divided by input count rate, ICR) is given by $\exp(-\tau*ICR)$ where, to a good approximation, the pulse-processing deadtime $\tau = 2T_p+T_f$ is the pulse basewidth. The dependence of resolution on peaking time is detector specific. Typical values of SLOWGAP are 6 for the 0-bit decimation FiPPI and 3 for the others. Different detectors may work best with slightly different values.

6.10.7 Setting the fast filter parameters(FASTLEN,FASTGAP)

In general the user does not modify these parameters directly, but through the host software routine SetAcquisitionValues (See Appendix).

The fast filter is also trapezoidal but has a decimation of 0 for all FiPPI designs. The values of FASTLEN and FASTGAP are given, for T_p' fast peaking time and T_f' = fast flat-top time in µsec:

$$FASTLEN = 40*T_p'$$

$$FASTGAP = 40*T_f'$$

Typical values of these parameters are FASTLEN=4 and FASTGAP=0. While these are reasonable values for most users, some may want to make FASTLEN larger in order to trigger at lower values (see below) or shorter to run at higher rates.

6.10.8 Setting the pulse detection parameter (THRESHOLD,MINWIDTH)

In general the user does not modify these parameters directly, but through a routine in the interface libraries.

X-rays are identified when the fast filter output (in units of ΔADC*FASTLEN) goes above threshold. This threshold can be expressed in energy units once the DXP conversion gain, G_{DXP} = number of ADC counts per keV at the DXP input, is known. For an energy threshold E_{th} in keV,

$$THRESHOLD=G_{DXP}*E_{th}*FASTLEN$$

The conversion gain is discussed below. The MINWIDTH parameter is used for noise rejection: It is the minimum number of time bins the fast filter is above threshold. A typivsl value that works with FASTLEN=4 is MINWIDTH=4.

6.10.9 Setting the Pile-up inspection parameters(MAXWIDTH,PEAKINT)

In general the user does not modify these parameters directly, but through the interface libraries.

MAXWIDTH is used to reject pulse pile-up on a time scale that is comparable to FASTLEN. The idea is that wide pulses are really two or more normal width pulses that are separated in time, but not enough for the fast filter to go below threshold. A typical value is

MAXWIDTH = 2*FASTLEN + FASTGAP +N

where N is in the range 4-8. If the signal rise-time depends on the x-ray energy (e.g. bandwidth limited preamplifier or low field regions of the detector that are preferentially sampled at some energy) this cut can bias the spectrum if it is too small.

PEAKINT is used to reject pulse pile-up when the pulses are well resolved by the fast channel. This value should be set as:

PEAKINT = SLOWLEN + SLOWGAP + N

where N = 2 or 3.

6.10.10 Setting the Gain

In general the user does not modify this parameter directly, but through the interface libraries.

The DXP internal gain is chosen to set the ADC dynamic range appropriately for the signals of interest. If it is set too low, the energy resolution may be compromised, while if is set too high there may be excessive deadtime. The ADC range is one volt full scale. Two guidelines are suggested for the internal gain setting:

1) This is appropriate when there is a single peak of interest: Set the gain such that the typical pulse height is between 4 and 10% of the ADC range (for a 10 bit ADC; or between 2 and 10% of the ADC range for 12 bit ADC).
2) This is appropriate when looking at a fixed energy range, with no particular peak of interest: Set the gain such that the maximum energy pulses are around 300-400 ADC channels (30-40% of the ADC range).

The parameter GAINDAC sets the internal amplifier gain. The overall gain can be expressed as follows:

Gtot = Gin * Gvar * Gbase

where

Gin : input buffer gain = 1 or ¼ depending on setting of jumper JPx00 (x=1, 2, 3, 4)
Gvar : variable gain setting = 1 to 100 depending on GAINDAC setting
Gbase : baseline gain ~ 0.5

Overall, the internal gain can range from about .5 to 50 V/V.

The gain control is a 16 bit DAC which sets the gain of a variable gain amplifier which is "linear in dB". The gain setting accuracy is approximately one bit (or 0.00061 dB = 0.007%). The relationship between Gvar and GAINDAC is:

Gain (in dB) = (GAINDAC/65536) * 40 dB

Gvar = 10**(Gain(in dB)/20)

In addition to the programmable gain control, a jumper (JPx00, located near the input relay for each channel) chooses an input stage gain of either 1 or ¼.

6.11 DSP Program Variants

MCA acquisition with pulsed reset preamplifiers (variant r01)

Variant 0 is the standard firmware variant supplied with the DXP-2X, as described in this manual. It is intended for use with reset preamplifiers (described in Section 3).

Firmware files:

Program file name: D2XR01nn.HEX, nn = version

Fippi file names: F01X2PxG.FIP (x=0,2,4,6)

Note: To use this variant, the "Ramp/Offset" jumper should be in the "Ramp" position.

MCA acquisition with resistive feedback preamplifiers (variant f01)

This firmware variant is intended for use with resistive feedback preamplifiers.

Firmware files:

Program file name: D2XF01nn.HEX, nn = version

Fippi file names: F02X2PxG.FIP (x=0,2,4,6)

Additional parameters (described in Section 0):

TAURC	Exponential decay time in 50 ns units.
RCFCOR	Correction factor (calculated automatically at start of run if TAURC not 0)

Note: To use this variant, the "Ramp/Offset" jumper should be in the "Offset" position.

This page intentionally blank.

7. Data Collection:

7.1. Overview:

In order to collect data, the spectrometer has to be configured, appropriate parameters downloaded to the DSP, a collection run started, the run concluded, and then data uploaded from the DSP to the host computer. Configuring the DXP was the topic of § 4 and 5. Selecting appropriate control parameters to control the run was discussed in § 6. In this section we will describe loading those parameters to the DSP, starting and stopping a run, and uploading data from the DSP. In § 8.5 we will then discuss how to use the run statistics to correct the spectral data for livetime and deadtime.

7.2. Setting Up for a Run:

Control parameters can be loaded either singly or in blocks. In brief, set the channel select bits in the GCR (General Control Register), write the target address to the channel's TSAR (Transfer Start Address register), then transfer the data. Programmers should refer to Appendix B for details. The parameters that are set for a given run would primarily set the gain and choose the peaking time, but other aspects of the processing are controllable as well.

While it is convenient to be able to change a single run parameter in order to gauge its effect on data quality, it is generally more convenient to be able to download an entire block of parameter values developed for data collecting under a particular set of circumstances. Thus, for example, if one wishes to take Cu K_α EXAFS spectra at high count rates, one might download the "CuK_Hi" parameter file in order to obtain a 1 μs peaking time and appropriate ASC gain values. The utility for developing libraries of parameter files for specific data collecting conditions is so high that most DXP Host Software should have this feature.

Another aspect of setting up for the run is setting the desired value of the RUNTASKS parameter. Each bit of RUNTASKS controls a particular function, as described in the previous section.

7.3 Controlling the Run Time':

It is important to note that the DSP data collection is typically started by writing to the GCR with the RunEna bit set to 1 and then stopped by writing to it again at a later time with RunEna set to 0. The DSP code checks the value of this bit after processing each event or baseline sample. If it detects an end of run, it disables the event interrupt, updates the run statistics and then returns to an idle state, waiting for further CAMAC commands. **Note**: it is therefore important, having sent a command to stop data collection, for the Host Software to pause to allow the DSP both to receive the signal and finish its data collection processing. 1 ms should be adequate. To be safe, one should check the value of the parameter BUSY; a value of 0 indicates that the DSP is in the idle state and the data is ready to be read out.

7.3.1. Using Host Software Control:

This is the simplest approach to controlling data collection time. The Host Software issues a "start collection" command, waits the desired collection period, and then issues a "stop collection" command. Because of latencies and delays in both the execution of the Host Software commands and the DSP, only approximate collection times will be obtained.

Note: In most cases this is not a problem at all because the DSP accurately records the precise amount of time (LIVETIME) it was collecting data and this value can be used to normalize the collected number of counts to a high degree of accuracy. The length of the run in real time (REALTIME) is also recorded; note that REALTIME does not reflect any time that datataking is disabled using the GATE signal. It is important to recall that the DXP does not collect data continuously from start of run to end of run in any case if for no other reason than that it cannot do so during preamplifier resets (nor can any other spectrometer, for that matter). In practice it will also spend small amounts of time assuring that the ASC's sawtooth function generator is correctly tracking the preamplifier input.

7.3.2. Using External Gate Control:

In certain conditions it may be useful to assure that all the channels on multiple DXP modules are collecting data during exactly the same time period. The DXP's "Gate" input (NIM or TTL) allows this to happen, since the FiPPI will only capture x-ray events (and let its livetime counter run) when this input is high. Since the Gate has an internal pull-up resistor, this is its default condition. An external logic signal can be applied to override this default. For example, if multiple modules were to be synchronized by a CAMAC Real Time Clock, the Host Software would first tell all DXP modules to start collecting data. Then it would start the Real Time Clock for the desired interval. Then it would issue stop collection commands to all the modules and proceed to read them out.

Reminder: Gate is High to enable data collection, Low to suppress it. When using the DXP with the external gate connected to a clock source, the DXP can be run in standalone mode (ignoring the external gate logic) by setting the "IgnoreGate" bit in the GCR, as described in Appendix B.

7.4. Common Retrieved Values:

Following a run the following data can be read from the DSP:

7.4.1. Error information:

ERRINFO; RUNIDENT; RUNERROR;

These parameters describe the run's termination status. For a successful run, RUNERROR will be 0. ERRINFO carries further information in case of a run terminated by a DSP error. See the DSP Software Manual [Ref. 1] for further details. RUNIDENT is a parameter that can be stored in memory to track a sequence of runs.

7.4.2. Spectral Data:

Spectral Data: up to 8k (8192) 24 bit words in program memory

These data contain the collected x-ray spectrum. The spectrum contains 8192 words 24 bits long. The offset of the MCA spectrum within program memory is stored in the parameter SPECTSTART. The parameters MCALIMLO and MCALIMHI indicate the actual length of the spectrum (MCALIMHI – MCALIMLO + 1?). Note that if MCALIMLO is set to be nonzero, an offset spectrum can be collected.

7.4.3. Event Related:

LIVETIME0,1,2; EVTSINRUN0,1; OVERFLOWS0,1; UNDRFLOWS0,1; FASTPEAKS0,1; REALTIME0,1,2

These values describe the data collected. LIVETIME and REALTIME are 48 bit quantities, with word 1 the least significant, word 0 in the middle, and word 2 the most signioficant (this numbering is historical). All of the other statistics are 32 bit quantities, with word 0 the high order word. LIVETIME contains the total time during which the FiPPI collected x-ray counts, measured in units of 400 ns (system clock divided by 16). REALTIME is the elapsed wall clock time, also in units of 400 ns. FASTPEAKS is the total number of discrete x-ray pulses detected in the FiPPI fast channel. Note that an x-ray pulse detected as having fast pileup is still only counted as a single event in FASTPEAKS. This allows the paralyzable dead time formula to be applied accurately. EVTSINRUN is the total number of counts binned into the spectral data. UNDERFLOWS and OVERFLOWS are, respectively, the number of good (not piled up) events which lie either below or above the binned energy range in the spectrum. Thus UNDERFLOWS should be normally equal zero when MCALOWBIN is zero.

Note: the total number of Good_Peaks (i.e. detected x-rays which are not piled up in either the fast or slow FiPPI filters) is thus the sum of events, underflows and overflows:

$$\text{Good\_Peaks} = \text{EVENTSINRUN} + \text{UNDERFLOWS} + \text{OVERFLOWS}. \quad (8.1)$$

7.4.4. Baseline Related:

BASEEVTS0,1; BASEMEAN0,1;

Baseline Histogram: 1024 16 bit words starting at address BASESTART in data memory.

Baseline history: A circular buffer containing the most recent baseline values starting at address HSTSTART in data memory.

The baseline data consist of both a spectrum of baseline events, stored in the indicated data memory area and some statistical values. The baseline spectrum may be plotted as a diagnostic. When the system is working properly the baseline spectrum should be essentially Gaussian in nature since it represents the electronic noise spectrum of the preamplifier/ASC electronics. It is useful for the Host Software to compute its FWHM value as an indication of electronic noise.

The parameters are:

BASEEVTS: the total number of baseline events in the spectrum.

BASEMEAN: the last value of the updating mean baseline value. This is an exponentially decaying running sum of recent baseline measurements.

7.4.5. ASC Tracking Statistics:

NUMASCINT0,1; NUMDRDOS0,1; NUMDRUPS0,1; NUMRESETS0,1; NUMUPSETS0,1; NUMZIGZAG0,1.

These parameters report statistics on how often the ADC input signal (i.e. the amplified difference between the preamplifier signal and the sawtooth function generator signal) drifted out of the ADC's input range and in what manner. These excursions may be due to: 1) preamplifier resets (which are large downward steps); 2) upward or downward drifts due to statistical fluctuations in counting rate; 3) large upward steps (caused, for example by cosmic ray detections); 4) Other, combination cases, such as reset followed closely be a cosmic ray. When the slope generator is correctly set in the sawtooth function generator the number of upward and downward out-of-range drift events should be approximately equal, since the generated slope will then match the average preamplifier slope.

The parameters are:

NUMASCINT0,1: the total number of out of range interrupts.

NUMRESETS0,1: the number of preamplifier reset event detected.

NUMDRDOS0,1: the number of downward drift events.

NUMDRUPS0,1: the number of upward drift events.

NUMUPSETS0,1: the number of upward jump (upset) events.

NUNZIGZAG0,1: the number of combination (zig-zag) events.

7.5. Livetime and Dead Time Corrections:

The theory and general practice of livetime and dead time corrections were discussed in § 2.7 and 2.8.

Since the DXP reports LIVETIME, this correction is trivial. Simple divide counts by livetime to obtain normalized counting rates.

Assuming that both the FiPPI slow channel and fast channel throughput curves have been measured and deadtimes obtained, as discussed, then the deadtime correction proceeds as follows.

1) Compute OCR: For the slow channel:

$$OCR = (EVENTSINRUN + OVERFLOWS + UNDERFLOWS)/LIVETIME \quad (8.2)$$

2) Compute ICR_t from FASTPEAKS by inverting the transcendental equation:

$$ICR_m = ICR_t * \exp(- ICR_t \ \tau_{df}), \quad (8.3)$$

where τ_{df} is the fast channel dead time, and the measured input rate ICR_m = FASTPEAKS/LIVETIME.

3) Scale the counts in the individual spectrum bins, N_i, by the ratio of ICR_t/OCR to obtain the "true" estimate of the counts in the bins:

$$N_{it} = N_i * ICR_t/OCR \quad (8.4)$$

After correcting for both fast and slow channel dead times, the DXP's integral count rate nonlinearity is typically less than one percent, and its the differential linearity over an order of magnitude less than that.

This page intentionally blank.

Appendix A: Release Notes:

As noted at the top of each page, this manual is version 0.9 for the second generation digital X-ray Processor, the DXP-2X. It is based largely upon version 1.3 of the DXP-4C manual.

. It corresponds as closely as possible to a specific release of the firmware, as follows:

DXP-2X Board revision:	B
Interface PROM revision:	1.0
DSP Code revision:	03
FiPPI Firmware revision:	G

We very much appreciate it if users notify us of problems with documentation, particularly in cases of errors or unclear explanations. You can reach us either by e-mail to support@xia.com, or by calling us directly.

This page intentionally blank.

Appendix B: CAMAC Interface Description:

B.1 Supported CAMAC operations

The DXP is controlled through a CAMAC interface, implemented in a Xilinx FPGA. The actions used to control the DXP, as described in the main text, require a relatively small number of CAMAC commands. The following CAMAC commands are supported:

CAMAC control operations:

Table B.1: CAMAC commands supported by the DXP module

F	A	Description
0	0	Read data from DSP memory (one channel).
1	0	Read General Control Register (GCR).
1	2	Read General Status Register (GSR).
1	5	Read Version Register
1	6	Read Timing Control Register (TCR; Model T only)
1	7	Read Timing Prescale Register (TPR; Model T only)
5	0	Level 1 FASTCAMAC Read from DSP memory
8	0	Test LAM. Returns Q=1 if LAM is set.
10	0	Clear LAM.
16	0	Write data to DSP memory (one or all channels).
17	0	Write to GCR (all bits)
17	1	Write Transfer Start Address Register (TSAR).
17	3	Download configuration to FiPPI.
17	4	Write to GCR: Run control bits *only* (Bits 0, 1; see below)
17	5	Write to GCR: Channel select bits *only* (Bits 6, 7 and 8; see below)
17	6	Write to TCR (Model T only)
17	7	Write to TPR (Model T only)
24	0	Disable LAM.
26	0	Enable LAM.
27	0	Test LAM source. Returns Q=1 if error bits are set internally.

The CAMAC convention denotes a command with F code of x and A code y by F(x)*A(y). All the above commands return the status bit X=1 (valid command), if both the F and A codes are correct. Unless otherwise noted, they also return Q=1 (command completed). The exceptions to this are F(8) and F(27), which return Q=1 only if the LAM is set.

CAMAC common control operations:

Reset (Z) Reset status register, disable LAM.

The CAMAC LAM interrupt can be used to notify the host machine when something is wrong. An example might be: if a DSP channel encounters a "serious" problem with data acquisition, such as an intolerably high event rate. To date the host software has not been developed to take advantage of this feature.

B.2 Registers Internal to the CAMAC Interface:

The General Control Register (GCR):

The General Control Register (GCR) is the main control register for all channels on the module, and uses a number of bits to determine what the DSP channels should do. The GCR is written using the command F(17)*A(0) and is read with the CAMAC command F(1)*A(0). All bits are both readable and writeable unless otherwise specified; write-only bits read back as zero. The GCR bits are defined in the following table:

Table B.2: General Control Register (GCR) bit assignments and indicated actions

Bit	Function
0	RunEna (0:stop, 1:start)
1	ResetMCA (0: no; 1: yes)
2	Reserved
3	LAMEnable
4	DSPReset (0: null; 1: reset DSP) (Write only)
5	LCAReset (0: null; 1: clear FiPPI) (Write only)
6-7	Channel # for next transfer
8	Broadcast (0: one channel; 1: all channels)
9	Reserved
10	Enable Self Reset
11	Ignore Gate (0:Gate enabled; 1:disabled)
12	Ignore Channel 0
13	Ignore Channel 1
14	Ignore Channel 2
15	Ignore Channel 3

On power up or CAMAC reset, all the GCR bits are cleared. The meaning of the individual bits, when written from CAMAC, is as follows:

RunEna: This bit determines whether data acquisition is active. When the DSP channels see this set, they proceed from the IDLE state to the data acquisition state. Likewise, when the bit is cleared, they proceed to the IDLE state. This bit can be cleared automatically at the end of fixed length runs if the *Enable Self Reset* bit is set (see below).

ResetMCA: If this bit is set, the DSP clears previously acquired data and statistics before starting the new run. If a run is started with this bit cleared, the DSP treats the new run as a continuation of the previous run, and newly acquired data are combined with the previously collected data. Note that in this case, the DSP assumes that no settings (gain, peaking time, etc) have been changed since the previous run. Any change in settings only take effect with the start of a new run (ResetMCA set).

LAMEnable: This bit determines whether LAM interrupts are enabled. Currently, LAMs are generated when any DSP enters an error condition. This bit can also be set [cleared] by the CAMAC command F(26)A(0) [F(24)A(0)].

DSPReset: If this bit is set when writing to the GCR, the selected DSP processor(s) will be reset, initiating a boot sequence. In general, this should be followed by writing the DSP code into program memory. (Write only, self clearing)

FiPPIReset: If this bit is set, the selected FiPPI channel (or all if Broadcast is set) will have its configuration cleared. In general this should be followed by a block write (F(17)*A(3)) to reload the configuration. (Write only, self clearing)

EnableSelfReset: If this bit is set, the RunEna bit is cleared internally at the end of a fixed length run, as soon as *all* enabled channels finish their run. To ignore a given channel, set the corresponding *Ignore* bit in the GCR (see below).

IgnoreGate: If this bit is set in the GCR, the external gate input is overridden (no events are blocked). This is used for operating the module in standalone mode in cases where the gate input is connected to an external gate source.

Ignore Chan n: These bits are used to mask off the various status bits (Active, FCErr, DSPErr) from the Global Status Register for unused channels. DSP errors from ignored channels do not trigger a LAM, and the run state (active) of ignored channels does not affect the self reset of the RunEna bit (if enabled).

The General Status Register (GSR):

The General Status Register (GSR) reports the current status of each channel (active, FiPPI error, DSP error), as well as a couple of global status conditions. The GSR is read with the CAMAC command F(1)*A(2); it is a read-only register. The GSR bits are defined in the following table:

Table B.3: General Status Register (GSR) bit assignments

Bit	Meaning
0	Channel 0 active (1: running, 0: idle)
1	Channel 1 active
2	Channel 2 active
3	Channel 3 active
4	LAM state (1: set, 0: clear)
5	Gate/Sync Logic Levels (0:TTL, 1: NIM)
6-7	Reserved
8	FiPPI Configuration Error Channel 0
9	FiPPI Configuration Error Channel 1
10	FiPPI Configuration Error Channel 2
11	FiPPI Configuration Error Channel 3
12	DSP Error Channel 0
13	DSP Error Channel 1
14	DSP Error Channel 2
15	DSP Error Channel 3

Note that if the Ignore Channel bit is set in the GCR for a given channel, the corresponding status bits for that channel (Active, FiPPI error, DSP error) will all read zero.

The meaning of the bits are as follows:

Active: If set, this bit indicates that the corresponding channel is running (actively taking data). If self-reset is enabled (bit 10 in the GCR), the RunEna bit in the GCR is cleared once all non-ignored (see bits 12-15 of the GCR) channels finish their fixed length runs (in this case, the host would not need to clear the RunEna bit). If this bit is clear, the corresponding channel is not running.

LAM State: This bit indicates whether there is a pending LAM (Look-At-Me) condition in the DXP. Currently, the LAM condition indicates that one or more DSP's are in an error state and require host intervention.

Logic Level: This bit indicates what logic level is set for the digital I/O lines on the front panel (Gate, Sync and Aux, where the latter two are only included in the timing model). If this bit is set, NIM logic levels have been selected; otherwise, the logic levels are TTL.

FiPPI Config Error: If set, this bit indicates that the FiPPI for the corresponding channel either has not been programmed or that the FiPPI download has failed.

DSP Error: If set, this bit indicates that the DSP for the corresponding channel has encountered an error condition, usually resulting from a calibration failure. See the DSP Code description chapter for further information on the possible error conditions.

The Transfer Start Address Register (TSAR)

The Transfer Start Address Register (TSAR) is used to specify the starting address in the specified DSP('s) for the next CAMAC data transfer. Note that these registers (one per DSP) reside in the DSP itself, and are not readable via CAMAC. The memory in each DSP covers a 15-bit address space; bit 14 is cleared to access the 16k words of 24-bit program memory, and is set to access the 16k words of 16-bit data memory. The target DSP is identified by bits 6 and 7 in the GCR (or bit 8 for a broadcast write); these bits must be set prior to writing to the TSAR. The TSAR is written with the command F(17) *A(1).

The Timing Control Register (TCR)

See Appendix E for a description of the Timing Control Register.

The Timing Prescale Register (TPR)

See Appendix E for a description of the Timing Prescale Register.

CAMAC data transfers to or from DSP Memory:

Note: Unlike the earlier model DXP-4C, data can be transferred to and from the DSP's in the DXP-2X module at any time, even during data taking. However, care should be taken not to overwrite parameters in the middle of a run, as results will be unpredictable. Intermediate spectra can be safely read from the module during a run (without pausing the run).

CAMAC block or single word transfers are used to read and write data to any location in DSP memory for each channel. Data transfers are done in three steps:

1) First, the target channel is identified by writing to bits 6 and 7 of the GCR. Alternatively, for a broadcast write (to all channels), bit 8 of the GCR is set (broadcast reads are not possible). Either the Write GCR command F(17)*A(0) or the Write GCR Channel bits command F(17)*A(5) can be used.

2) The starting address for the upcoming data transfer to or from DSP memory is written to the Transfer Start Address Register (TSAR) using the F(17)*A(1) command.

3) The data are read from or written to the DSP memory using F(0), F(5) to read or F(16) to write, either with a single word transfer or a block transfer.

It is possible to write in parallel to (but not read from) all channels on a DXP board. This is accomplished by setting the GCR's "Broadcast" bit to 1 in step 2. The broadcast write is generally used to download the same parameters or configuration data to all the channels at once, as well as to download the DSP code into program memory.

DSP memory is split into two 16k blocks: program memory and data memory. Program memory is made up of 24-bit words; for the standard DXP application, it holds both the DSP program and the MCA

spectrum. Two 16-bit CAMAC data transfers per word are required when accessing program memory; bytes 1 and 2 are transferred first, and byte 0 is transferred as the lower byte of the second transfer (the high byte of the second transfer is ignored on a CAMAC write and is zero on a CAMAC read). Data memory is made up of 16-bit words, and requires only one CAMAC transfer per word.

There are two types of CAMAC reads from DSP memory: a standard read using F(0)*A(0), and a Level 1 FASTCAMAC read using F(5)*A(0). The standard read can transfer up to one word per microsecond (the length of a standard CAMAC cycle), for a peak transfer rate of 2 Mbyte/sec. Using the FASTCAMAC block read, one word is transferred every 400 ns, for a peak transfer rate of 5 Mbyte/sec. Briefly, this is accomplished by combining the entire block transfer into one CAMAC cycle, repeating the S1 strobe every 400 ns for each word transferred, and finally completing the cycle with a single S2 strobe. See the FASTCAMAC specification for more detailed information [ref to yale/fastcamac website].

Initiating Data Acquisition with the DXP:

After downloading the DSP code and the parameter values to choose the running conditions (gain, peaking time, etc.), data taking is initiated by setting the RunEna bit in the Global Control Register (GCR). This alerts the DSP's, causing them to proceed to the data acquisition loop. The DSP's event processing loop is described in detail earlier in this manual. For completeness, we list here the steps the DSP takes to begin data collection:

1) Check the ResetMCA bit in the GCR. If set, the DSP starts by clearing old data. If the ResetMCA bit is cleared, data taking begins immediately (skipping steps 2 and 3 below).

2) Write filter constants to the FiPPI and starts the FiPPI processing.

3) Checks to see if the analog settings (gain, polarity, etc) have changed; if so, the new values are written to the ASC, and any necessary calibrations are performed.

4) The Event interrupt is enabled and data taking begins.

5) Begin storing the x-ray events captured by the FiPPI into the MCA. Buffer events as they arrive from the FiPPI. As events are processed, calculate their pulse heights and increment corresponding bins in the MCA. When there is no event to process, collect baseline data and update the baseline average.

6) Each time around the loop, check the RunEna bit to see whether data acquisition is still enabled. If not, end the run and go to the idle state.

7) Every few hundred microseconds (on interrupt control), update the livetime and realtime counters.

8) If the ADC drifts out of range (due to a reset, for example), make the necessary adjustments to the ASC to bring the signal back in range.

For a fixed length run, when the DSP has finished collecting the requested number of x-ray events, it proceeds to IDLE state, and waits for CAMAC to unload its MCA data. Alternatively, if the Host decides to end data taking, it would clear the RunEna bit in the GCR, which tells the DSP to stop the run.

This page intentionally blank.

Appendix C: Firmware Configuration:

On power up, the module needs to have its "configuration" (DSP code and Xilinx firmware) downloaded using CAMAC data transfers. In both cases, the data are read from ASCII files on the host computer. In general, the FiPPI firmware should be loaded first, then the DSP program. To switch FiPPI configurations once the DSP is running, it is necessary to put the DSP into a special 'dormant' mode (by running a special task; see Chapter ??), downloading the new FiPPI configuration, and then waking up the DSP.

FiPPI Configuration Downloading:

As described in Section 2??, there are four sets of FiPPI configuration data, corresponding to the four ranges of slow filter peaking times. These principally differ by the number of decimation bits: 0 for shortest peaking times (0.125 to 0.625 µsec), 2 for medium peaking times (0.5 to 2.5 µsec) , 4 for long peaking times (2.0 to 10 µsec) and 6 for very long peaking times (8 µsec to 40 µsec. These configuration file are denoted by names of the form FmmXPrst, where F indicates this is a FiPPI file, mm is the FiPPI variant, XP indicates this is an X-ray processor, r indicates the hardware board revision (currently B), s is a logic revision (A-Z) and t is the number of decimation bits (0, 2, 4 or 6).

The FiPPI logic is loaded using the FPGA's "Slave Serial Mode" to write 8 bits of data per CAMAC transfer cycle.

The sequence of events to download the FiPPI logic from the host computer is:

1) Read the configuration data for the appropriate FiPPI decimation design from the file on disk. This is an ASCII file with extension .FIP.
2) Write to the CSR with the LCAReset bit (bit 5) asserted, to initiate the process. Usually, bit 8 is also asserted so all channels are downloaded at once; otherwise bits 6 and 7 specify which channel to download.
3) Write the configuration data using a block transfer to F(17)*A(3).

DSP Program Downloading:

On power-up or reset, the DSP for each channel must be initialized and the DSP program must be downloaded. The code is written into DSP program memory using normal CAMAC writes into DSP memory space (using the command F(0)*A(0)). While the details of this procedure will normally be performed by either the MESA DXP Control software or by the driver routines provided with the DXP, the procedure is outlined below:

1) Read the image data and the symbol table for the desired DSP program from a file on disk. This is an ASCII file with extension .HEX which contains both the image and symbol table.
2) To initiate the process, write to the CSR with the DSPReset bit (bit 4) asserted. Bits 6 and 7 should specify which channel to download, or bit 8 should be asserted in order to download all channels at once.
3) Write words 1 through N (all except the first word) to program memory, starting at memory location 0x0001 (write 0x0001 to the TSAR, then do a block data transfer). Recall that two CAMAC data transfers are needed for each write to a 24-bit program memory location; bytes 1 and 2 are transferred first, then byte 0 is transferred last (as the lower byte of the second word transferred).
4) Write word 0 to program memory location 0 (write 0 to the TSAR, then write the data). This starts the DSP running.

As soon as the program is completely downloaded , the DSPs automatically initialize their constants in the DSP Parameter memory (the first few hundred words in data memory). They then go through some initial internal calibrations and proceed to the IDLE state, waiting for an instruction to start taking data.

This page intentionally blank.

Appendix D: DSP/FiPPI/ASC Communication and Control:

Communication between the DSP and other DXP sections is accomplished by having the DSP write to (or read from registers in the FiPPI and ASC which are set up as addresses in the DSP's external address space.

DESCRIPTION OF FIPPI REGISTERS GOES HERE

This page intentionally blank.

Appendix E: Timing Applications for the DXP-2X Model T:

The timing option in the DXP-2X Model T offers additional capability for experimental situations involving timing measurements. Typically, X-ray pulses are labeled with a "time" value and either kept in a list, sorted into different spectra, or their number stored as a function of time. Four of these applications are described below, though others are envisioned. Each would require minor changes to the DSP software algorithm, and potentially also to the FiPPI configuration logic. Since these firmware configurations are downloaded at run time, switching between the special and standard applications is easy. Users are encouraged to contact XIA to discuss other potential applications.

The timing functions in the DXP-2X Model T are controlled by two registers: the Timing Control Register (TCR) and the Timing Prescale Register (TPR). The TCR determines the basic functionality, such as clock source; the TCR is written with the command F917)*A(6) and read with the command F(1)*A(6). The TPR stores the prescale value, for the case where a prescaled clock is selected in the TCR; the TPR is written with the CAMAC command F(17)*A(7) and read with the command F(1)*A(7). The TCR and TPR contents are described in the following sections.

The Timing Control Register (TCR)

The contents of the TCR are described in Table E.1 below. The timer is distributed to the individual channels on the Isync line; the timer can be disabled by setting bit 0 in the TCR. Bit 1 in the TCR is used to enable interrupts on the rising edge of the GATE signal, which are needed for some timing applications (for example, the rising edge of GATE could indicate when to switch from one pixel to the next in a scanning application). Bit 2 of the TCR selects between using the direct clock (either internal or external) or a prescaled (divided down) clock.

Table E.1: Timing Control Register definitions:

Bit	Definition
0	Timer Disable (0: Timer enabled, 1: Timer disabled)
1	Enable Gate Interrupts (0: No interrupts, 1: Interrupt DSP on riding edge of GATE)
2	Prescale (0: Use direct clock, 1: Use prescaled clock)
3	Enable Synch Run: If set, synchronize run periods for multiple modules (see below)
4, 5, 6	Clock Select Bits 0, 1, 2. Selects between external clock and various internal clock rates
7	Enable Timer Output: If set, the timer is driven onto the Aux I/O connector
8-15	Reserved

The individual bits of the TCR are described below:

Timer Disable: If this bit is set, the timer signal is not driven onto the internal ISync line. This is of use when the timing module is used for non-timing applications (the standard application firmware will ignore the timer signal, but the system may run quieter if the signal is turned off).

Enable Gate Interrupts: If this bit is set, the DSP's are interrupted on each rising edge of the GATE signal. This bit must be set for timing applications that use the GATE signal to move between multiple spectra or SCA's. If this bit is cleared, no DSP interrupts are generated.

Prescale: If this bit is set, the input clock (either internal or external) is prescaled (divided down) by the value specified in the Timing Prescale Register. Very long time periods are achievable using the prescaled clock. If this bit is cleared, the source clock (specified by the clock select bits) is applied to the ISync line.

Enable Synch Run: If this bit is set, runs can be synchronized between multiple modules. Normally, since runs are begun or ended by a single CAMAC write, it is impossible to start or stop a run in two modules simultaneously (you can only address one module at a time over the CAMAC bus). In order to synchronize the runs, the Aux I/O port is used as a Busy output; the Busy outputs for all modules must be ANDed together, and the result fed back into the Sync input. When a module starts a run, it completes all the startup tasks, then sets the busy output; the actual data taking does not begin until the Sync line is asserted. Similarly, when the run is ended on one module, the run ends on all modules. Note that if this setup is used, the Sync line cannot be used as a source for the clock, and the timer cannot be driven onto the Aux I/O. Also note that it is possible to synchronize the data taking just by holding the GATE signal low until all channels are ready to take data.

Clock Select: The three clock select bits are used to choose between using the external Sync input as the source clock or one of seven clock speeds derived from the internal system clock. The available clocks are listed in **Table E.2** below:

Table E.2: Clock Selection:

TCR Bits 6, 5, 4	Clock Source
0, 0, 0	External Sync input (up to 40 MHz)
0, 0, 1	Internal clock CLK (40 MHz: 25 ns period)
0, 1, 0	CLK/2 (20 MHz: 50 ns period)
0, 1, 1	CLK/4 (10 MHz: 100 ns period)
1, 0, 0	CLK/16 (2.5 MHz: 400 ns period)
1, 0, 1	CLK/64 (625 kHz: 1.6 µs period)
1, 1, 0	CLK/256 (156.25 kHz: 6.4 µs period)
1, 1, 1	CLK/1024 (39.06 kHz: 25.6 µs period)

The Timing Prescale Register (TPR)

If the Prescale bit is set in the TCR (see above), then the clock (either internally or externally generated) is prescaled (divided down) based upon the value in the Timing Prescale Register (TPR). If a prescaled clock is selected and the value N is stored in the TPR, then the clock source is divided down by a factor of N+2 in order to produce the timing clock on Isync (for example, if 0 is stored in the TPR, then the clock rate is cut in half). Using the 16 bit prescale register and the wide selection of clocks, it is possible to generate almost any clock period ranging from 25 nanoseconds (with a 40 MHz direct clock) to 1.678 seconds (using the slowest clock and the largest prescale) or more if an external clock source is used.

Table E.3: Timing Prescale Register definition:

TPR Bits	Definition
0-15	Prescale value N. Clock divided by N+2 if Prescale bit is set in TCR.

General Description of the Timing Functionality

The DXP-2X Model T incorporates two additional front panel digital signals beyond the GATE signal in the Model C: the SYNC input and the AUX I/O port. The SYNC signal works in conjunction with the GATE input to enable data acquisition. The external GATE and SYNC signals connect to the module interface FPGA, as well as to the FiPPI FPGA on each channel, as shown in **Figure E.1**. The internal sync ISync can be used in a variety of ways, as described below. For example, it can be used to clock a counter within the FiPPI for time resolved acquisition, or it can also be used to reset a counter within the FiPPI for multiple spectra acquisition, or alternatively the DSP can read the level of the Sync input from the interface. **Figure E.1** is included for better understanding of how the major components of a DXP channel are connected, and also to suggest to the user alternate possibilities for using the Gate and Sync logic. The current uses of the Aux I/O port are described in the section on the TCR; this signal is available for timing applications beyond those described below. Note that both GATE and SYNC are inputs, and can be configured to be compatible with NIM or TTL logic using jumpers. The AUX I/O port is bidirectional for TTL logic, but only an output if NIM logic levels are selected.

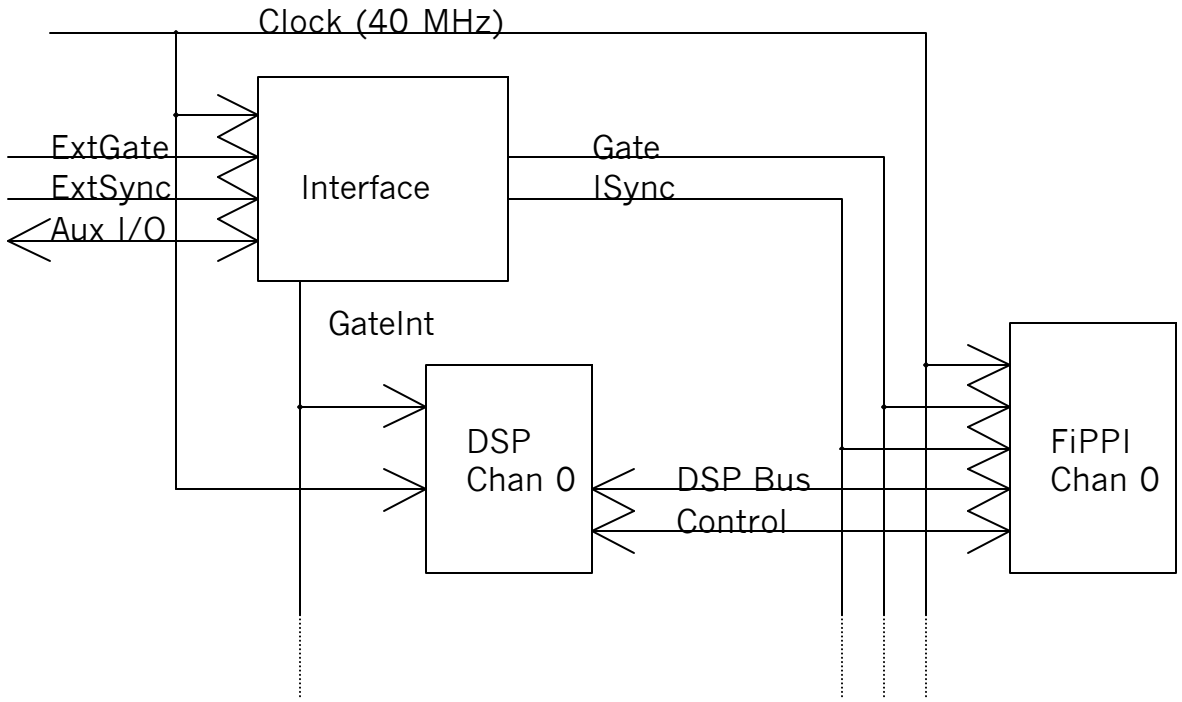


Figure E.1: Schematic block diagram of the major digital components of a DXP channel, along with the CAMAC interface FPGA.

Application 1: "Phase Locked" acquisition into 2 MCA spectra

This application has already been coded and tested, and has been designated as DSP Program Variant 4. It was first used by F.Bridges and C.Booth of University of California at Santa Cruz, as reported at the EXAFS IX Conference (1996). In their case, a sample was pulsed between normal and superconducting

several times a second, and the data collected separately into two MCA spectra, each up to 4k bins wide. The purpose of the spectrum switching is to remove low frequency sources of "noise" from the EXAFS data.

The Gate and Sync inputs are used as shown in the example figure below:

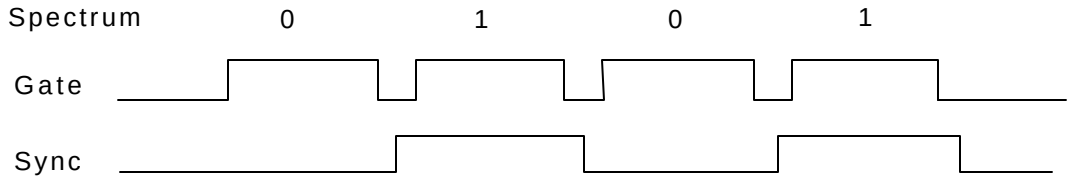


Figure E.2: Gate and Sync signal timing for phase locked acquisition into 2 MCA spectra.

On each low to high Gate transition, the DSP reads the level of the Sync pulse, to determine which spectrum will be collected. It then updates the live time and ICR count for the previous spectrum, and continues taking data as before. In this way deadtime corrections can be applied as accurately as for the standard single MCA spectrum.

Application 2: Acquisition of Multiple (more than 2) MCA spectra

Switching between multiple spectra would be a generalization of the 2 MCA spectra case. This application uses the leading edge of the GATE signal to indicate when to move to the next spectrum. If the SYNC signal is high on the rising edge of GATE, then the chosen spectrum is set to the first in the series.

The figure below illustrates an example with 4 MCA spectra. Data is taken during each period when the Gate input is high. The Sync input would be used to reset a spectrum counter in the DSP. As in application 1, the DSP would keep track of which spectrum was being accumulated, and update the livetime and ICR count appropriately. The time intervals would not need to be evenly spaced or of the same length.

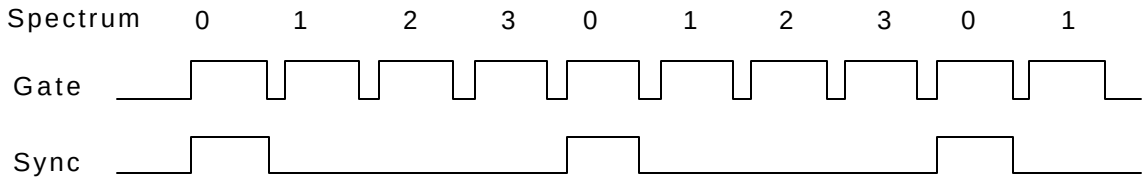


Figure E.3: Gate and Sync signal timing for multiple MCA spectra acquisition.

A small number of spectra could be fit in the standard 8k spectrum memory, with smaller numbers of bins per spectrum. For example, you could have 4 spectra of 2k bins each or 16 spectra of 512 bins. For a larger number of spectra, the DXP channels can be outfitted with 1 Mbyte of additional static RAM. This would allow, as an example, 256 spectra of 1024 bins each (using 32-bit bins), so one full row of a 256x256 scanning operation could be stored locally (read the data once per row).

Application 3: Multi-Channel Scaling (MCS): Time resolved acquisition of SCA windowed data

A third application is Multi-Channel Scaling (MCS) which is the collection of time resolved SCA data. The MCS mode can be used, for example, to look at the evolution of a fluorescence peak as a function of time after some stimulus, either in single shot or stroboscopic mode. It is currently implemented in DSP program variants 10 and 11, and does not require the extended memory option.

In this case, each event detected has a time associated with it, which would be the time since the rising edge of the Gate signal. As shown below, the Gate signal might be repeatedly asserted to indicate the time of the stimulus or impulse (or shortly before). The counter/timer within the FiPPI is reset by the rising

edge of the Gate, and counts pulses on ISync. The basic time granularity or "dwell time" is set by the ISync period. If the internal DXP clock is used, this ranges from 50 nsec and 0.82 msec; an external clock can be provided into the ExternalSync input, for longer time periods or greater timing accuracy. A16 bit timing counter programmed in the FiPPI counts the ISYNC pulses, allowing one to look at phenomena on a variety of timescales from a few hundred nanoseconds to about 50 seconds with the internal clock.

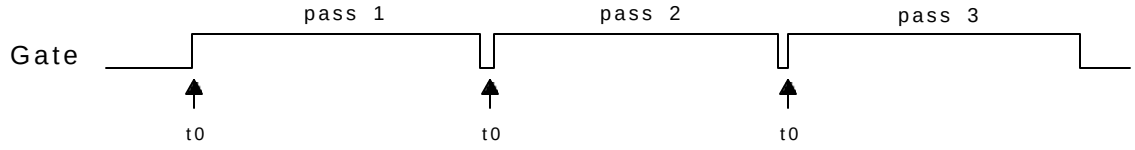


Figure E.4: Gate and Sync signal timing for multi-channel scaling acquisition.

The number of events within an energy window is recorded as a function of time after Gate rises (t0). The DSP program variant 10 implements the MCS in a single 1024 channel time distribution. If one desires to make pile-up dead time corrections, the total (unwindowed) count rate can also be recorded as a function of time, using program variant 11. This option has 512 time bins each for the windowed and total output counts. To correct for pulse pileup, one would scale each time bin by a correction factor (a function of output count rate) which would need to be empirically determined or calculated.

Users interested in further MCS options are encouraged to contact XIA.

Application 4: List mode acquisition of Time resolved MCA data

A fourth application is the collection of time resolved MCA data in a list mode. In this case, a list of pulse heights and time after a Gate signal assertion is accumulated, where the Gate signal is as shown in the Figure E.4 above. When the list memory becomes full (or nearly full), the channel can interrupt the host processor with the CAMAC LAM, or the processor can poll the DXP to check if the LAM is set. The accumulated list of events and times is then read out by the host processor and analyzed (binned into MCA spectra) there. The list mode acquisition is currently implemented in DSP program variant 20, and uses the additional memory option DXP-4M.

In principle, the list might also include "pseudo-events" such as the time of starting and stopping data acquisition, for example due to pre-amplifier resets and when the host processor is reading out. In this way the live time when data is taken could be corrected for as a function of time after Gate assertion. This feature is not currently implemented; users interested in this or other list-mode options should contact XIA.