Basic description of the KETEK DPP3 low-level communication

In the following general information regarding the DPP3 low-level command set, user data concept, and firmware update functionality will be given.

1 Contents

1. DPP3 command set

In this chapter the general commend set of the DPP3 using low-level communication will be described. The command set is equal for each interface of the DPP3 (Ethernet, USB, and SPI). All data are transmitted from most significant to least significant bit (MSb first). Please note that in case of SPI communication application data (requests to the DPP3 and responses from the DPP3) are exchanged using the SPI protocol described in the document “DPP3_SPI_Description”.

1.1. Parameters of the DPP3

The documenet “DPP3_Complete_Parameter_Set_Overview” contains all parameters of the KETEK DPP3. The columns of the sheet are described in the following:

1.1.1. Parameter ID

Each parameter is identified using a unique number which is called parameter ID. The listed values are in decimal representation.

1.1.2. Parameter name

The name of the parameter.

1.1.3. Parameter description

Brief description of the parameter.

1.1.4. Type

The type of the parameter. Three types of parameters are distinguished:

- Read/Write (“R/W”): For parameters of this type read and write operations are possible. Therefore, the current value of the parameter can be read out or a new value can be written by the user.
- Read only (“R”): For parameters of this type only read operations are permitted. Write operations will be rejected with an error code.
- Function (“Func”): Accessing parameters of this type triggers the execution of a certain action in the DPP (e.g., start a measurement). Read operations are meaningless since there are no values linked to these parameters.

1.1.5. Minimum write value

Minimum allowed value of write operations for the parameter. The listed values are in decimal representation. Trying to write a value below this limit will cause an error code in the response.

1.1.6. Maximum write value

Maximum allowed value of write operations for the parameter. The listed values are in decimal representation. Trying to write a value above this limit will cause an error code in the response.

1.1.7. Value description

Description how values for the parameter are interpreted.

1.1.8. Request syntax

Indication whether requests are constructed according to the standard syntax (see: 1.2.1 Standard structure of requests) or if there is a special syntax (see: 1.2.2 Special structure of requests) for the parameter.

1.1.9. Response syntax
Indication whether responses can be interpreted according to the standard syntax (see: 1.3.1 Standard structure of responses) or if there is a special syntax (see: 1.3.2 Special structure of responses) for the parameter.

1.2. Structure of requests
In the following, the syntax of requests to the DPP3 will be described.

1.2.1. Standard structure of requests
Most parameters of the DPP3 are accessible by the following structure. These parameters are noted with request syntax “standard”. The standard syntax consists of a 4-byte data frame:

Parameter ID	Command	MSByte of data	LSByte of data

The parameter ID is the unique number of each parameter (see: 1.1 Parameters of the DPP3) with a width of 1 byte. The command specifies the operation of the request and can be either 0x00 for a read operation or 0x01 for a write operation. For parameter of the type “func” arbitrary values for the command can be used. The last two bytes of the request contain the data of the request, which are only used for write operations and some parameters of the type “func”. For read operations arbitrarily data can be used in the request.

The user can send multiple requests in one transmission. Up to 32 data frames with 4 bytes each can be stacked. Stacked requests are processed serially in the DPP3 and in the same way as if they are transmitted individually.

1.2.2. Special structure of requests
Some parameters use a different syntax than the one described above. These parameters are noted with request syntax “Non-standard” in the parameter overview file.

ID 19 - MCA Read

Requests of the parameter “MCA read” (ID 19) use the same 4-byte structure as standard-parameters. However, stacked data frames are not permitted with this parameter. The parameter must be transmitted individually, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned.

ID 71 - MCU Passthrough

The parameter “MCU Passthrough” (ID 71) is used to transmit data to the MCU. The MCU can be controlled using datagrams described in the additional document “Definition_MCU_Datagram”. In order to communicate with the MCU the parameter ID 71, followed by the command 0x01, followed by the datagram is send to the DPP3:

Parameter ID	**Command**	**MCU datagram**		
71	0x01	Frist byte of datagram	…	Last byte of datagram

The DPP3 passes the datagram to the MCU, reads the response of the MCU and returns the response to the user. Stacked data frames are not permitted with this parameter. The parameter must be transmitted individually, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned.

ID 84 - Get Event Scope

Requests of the parameter “Get Event Scope” (ID 84) have the same 4-byte structure as standard-parameters. However, stacked data frames are not permitted with this parameter. The parameter must be transmitted individually, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned.

ID 92 - Write Firmware Section

The parameter “Write Firmware Section” (ID 92) is used to update the DPP firmware. Stacked data frames are not permitted with this parameter. The parameter must be transmitted individually, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned. The parameter can only be called successfully after “Delete Firmware” (ID 91) was requested, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned. After calling “Delete Firmware” (ID 91) every firmware section can be written one time. If the device is power cycled “Delete Firmware” (ID 91) has to be called again. The DPP firmware consists of 4096 sections (sections number 0 to 4095) with a length of 1024 byte each. In order to write a firmware section, the following request structure is used:

Parameter ID	**Command**	**Section number**		**Firmware data**		
92	0x01 (irrelevant)	Section number MSB	Section number LSB	First data byte of firmware	…	1024th data byte of firmware

Therefore, the request has a total length of 1028 byte. The response of the DPP3 will be a standard 4-byte data frame with the following structure:

Parameter ID	**Status code**	**Section number**	
92	Status code	Section number MSB	Section number LSB

The internal processing time of this parameter is higher (typically 1ms) than for most other parameters.

1.3. Structure of responses

1.3.1. Standard structure of responses

Most parameters of the DPP3 use this response structure. These parameters are noted with response syntax “standard” in the parameter overview. The standard structure consists of a 4-byte data frame:

Parameter ID	Status code	Data MSByte	Data LSByte

The parameter ID is the unique number of each parameter (see: 1.1 Parameters of the DPP3) with a width of 1 byte. The second byte of the response indicates the status of the request. The status byte is 0x00 if the request was processed successfully. Any other status byte indicates an error during request processing. The last two bytes of the response contain the data. The following table contains all status codes that may be returned together with the corresponding interpretation of data bytes:

Status code	**Description**	**Interpretation of data bytes**
0x00	Request was processed successfully	Valid parameter value

0x01	Error: Requested value is out of range and cannot be applied	Closed valid value for requested parameter and value
0x02	Error: Requested parameter can only be read, write request is declined	0x0000
0x03	Error: Requested parameter does not exist	0x0000
0x04	Error: Incorrect command byte	0x0000
0x05	Error: Requested parameter can currently not be accessed (e.g., measurement running, DPP in power down)	0x0000
0x06	Error: DPP-internal timeout occurred	0x0000
0x07	Error: Unexpected length of data	0x0000
0x08	Error: Incorrect syntax of request	0x0000

If multiple requests were sent in one transmission the responses are also send in one transmission and stacked equivalent to the requests.

1.3.2. Special structure of responses

Some responses use a different syntax than the one described above. These parameters are noted with response syntax "Non-standard" in the parameter overview file.

ID 18 - Runtime Statistics Read

The response for "Runtime Statistics Read" (ID 18) has a length of 13 x 4Byte with the following structure:

Parameter ID	Status code	Parameter value MSB	Parameter value LSB
5	0x00	MSByte of "Run Active" (ID 5)	LSByte of "Run Active" (ID 5)
6	0x00	MSByte of "Realtime Low" (ID 6)	LSByte of "Realtime Low" (ID 6)
…			
17	0x00	MSByte in „Output Count Rate High" (ID 17)	LSByte in „Output Count Rate High" (ID 17)

Therefore, this function is used for synchronous readout of the parameters ID 5 to 17 (see parameter overview document "DPP3_Complete_Parameter_Set_Overview". This is useful if the user wants to calculate further statistics with the values (e.g., deadtime ratio) and wants to avoid time delays due to successive readout of the values.

ID 19 - MCA Read

The parameter "MCA Read" (ID 19) is used to read out MCA data from the DPP3. The response consists just of the MCA data (no header) and the size is determined by the current settings of "MCA Number of Bins" (ID 20) and "MCA Bytes per Bin" (ID 21). The counts of each bin are transferred with the LSByte first. Transmission begins with bin 0.

Example: Parameter ID 20 „MCA Number of Bins" is set to 13 and parameter ID 21 „MCA Bytes per Bin" is set to 2. "MCA Read" returns 2^13 x 2 byte = 8192 x 2 byte = 16384 byte. The first byte of the data is the LSByte of Bin 0 and the last transferred byte is the MSByte of bin 8191:

MCA data

LSByte of bin 0	MSByte of bin 0	LSByte of bin 1	MSByte of bin 1	…	LSByte of bin 8191	MSByte of bin 8191

ID 71 - MCU Passthrough

The parameter "MCU Passthrough" (ID 71) is used to transmit data to the MCU. The response consists of a header with a length of 2 byte, followed by the response of the MCU. The header contains the following information:

Parameter ID	**Status code**
71	Status of passthrough process

The second byte of the header indicates the status of the UART transmission:

0: Success
1: UART Timeout (no MCU response received)
2: DPP-internal memory error
3: Incomplete datagram was requested

ID 79 - Read all parameters

The parameter "Read all parameters" (ID 79) can be used to read out the entire parameter memory of the DPP3 with one request for diagnosis purposes. The response consists of 256 x 4 byte with the following structure:

Parameter ID	**Status code**	**Parameter value MSB**	**Parameter value LSB**
0	0x00	MSByte of parameter ID 0	LSByte of parameter ID 0
1	0x00	MSByte of parameter ID 1	LSByte of parameter ID 1
…			
255	0x00	MSByte of parameter ID 255	LSByte of parameter ID 255

For unused parameter IDs and for parameters of the type "func" the data 0x0000 will be returned.

ID 84 - Get Event Scope

The parameter "Get Event Scope" (ID 84) is used to read out event scope signal data from the DPP3. Each scope contains 8192 signal values with a width of 3 byte each. The response consists only of the signal data (no header with parameter ID) and the size therefore is 8192 x 3 byte = 24576 byte. The first byte of the data is the LSByte of the first signal value and the last transferred byte is the MSByte of the last signal value.

ID 91 – Delete Firmware

The Parameter "Delete Firmware" (ID 92) deletes the so-called update image of the DPP firmware (see 3.1 Firmware memory concept). The syntax for requests and the structure of responses equal standard parameter. However, the time needed for internal processing of the request is very high (typically 30s, maximum 90s).

ID 93 - Read Firmware Section

The parameter "Read Firmware Section" (ID 93) is used to read out the update image of the FPGA firmware. Stacked data frames are not permitted with this parameter. The parameter must be transmitted individually, otherwise a status-code 0x08 (see: 1.3.1 Standard structure of responses) will be returned. The FPGA firmware consist of 4096 sections (section number 0 to 4095) with a length of 1024 byte each. In order to read a firmware section, a request with standard syntax is sent:

Parameter ID	**Command**	**Section number**	
93	0x00 (irrelevant)	Section number MSB	Section number LSB

The response has a total length of 1028 byte, in which 4 bytes are a header and 1024 bytes are firmware data:

Parameter ID	**Status code**	**Section number**		**Firmware data**		
93	Status code	Section number MSB	Section number LSB	Frist data byte of firmware	…	1024th data byte of firmware

ID 127 - Force EOL

The parameter "Force EOL" (ID 127) is only used internally in the DPP3 for controlling the communication process. There is no relevant function of this parameter for the user. No response will be sent by the DPP3 if this parameter is called by the host.

1.4. Example

In this example the application data for a simple measurement process will be shown.

1.4.1. Configuration of the DPP3:

First, the parameter values of the DPP3 are set to desired settings for a certain application. Commonly set parameters are for example:

Slowfilter Peaking Time

"Slowfilter Peaking Time" (ID 36) is chosen depending on the count rate as well as on desired energy resolution and X-ray pulse throughput. In this example the peaking time will be set to 100ns. According to the parameter overview sheet this corresponds to a parameter value of 100ns/12.5ns = 8. Therefore, we write the value of 8 to the parameter ID 36. Since "Slowfilter Peaking Time" (ID 36) has the standard request syntax, a 4-byte data frame is needed:

36	1	0	8

Since "Slowfilter Peaking Time" (ID 36) has the standard response syntax, a 4-byte data frame will be received from the device:

36	0	0	8

The second byte indicates the status (see 1.3.1 Standard structure of responses). The user should verify that a 0 is returned, which means the request was processed successfully. In this case the peaking time has been set to 100ns.

Fastfilter Trigger Threshold

“Fastfilter Trigger Threshold” (ID 38) is chosen depending on the present electronic noise in the system and on the desired performance at low X-ray energies. A typically used value might be 80. Therefore, we write the value of 80 to the parameter ID 38. Since “Fastfilter Trigger Threshold” (ID 38) has the standard request syntax, a 4-byte data frame is needed:

38	1	0	80

Since “Fastfilter Trigger Threshold” (ID 38) has the standard response syntax, a 4-byte data frame will be received from the device:

38	0	0	80

The second byte indicates the status 0, which means the request was processed successfully. The trigger threshold of the fastfilter was set to 80.

Run Stop Condition

“Run Stop Condition Type” (ID 2), “Run Stop Condition Value Low” (ID 3), “Run Stop Condition Value High” (ID 4) are chosen depending on how long the measurement of the DPP should run. In this example we will configure the DPP3 to stop the measurement after 120s of realtime. Therefore, we write the value of 2 to the parameter ID 2 (stop at fixed realtime). According to the parameter overview sheet 120s corresponds to a parameter value of 120s/10µs = 12000000 = 0x00B7 1B00. Therefore, we write the value 0x1B00 to the parameter ID 3 and the value 0x00B7 to the parameter ID 4. Since all three parameters use the standard request syntax, for each a 4-byte data frame is needed. The three data frames can either be transmitted individually (each with a transmission with 4-byte application data) or in one transmission (one transmission with 12 byte):

2	1	0	2
3	1	0x1B	0x00
4	1	0x00	0xB7

Since all three parameters have the standard response syntax, a 4-byte data frame will be received from the device for each request

2	0	0	2
3	0	0x1B	0x00
4	0	0x00	0xB7

The second byte of each data frames indicates the status 0, which means the request was processed successfully. The stop condition was set to 120s fixed realtime.

1.4.2. Start of the run

After all parameter values has been set to desired values, the measurement run can be started. This is done using the parameter “Run Start” (ID 0). According to the parameter overview sheet a value of 0 in the request deletes existing MCA data (new run), while a value of 1 adds new data to the existing MCA data (resume run). In this example we delete existing MCA data and therefore set the data bytes to 0. The parameter type is “func” and therefore the command byte can be chosen arbitrary. Since “Run Start” (ID 0) has the standard request syntax, a 4-byte data frame is needed:

0	0	0	0

Since “Run Start” (ID 0) uses the standard response syntax, a 4-byte data frame will be received from the device:

0	0	0	0

The second byte indicates the status 0, which means the request was processed successfully. The measurement now runs on the DPP3.

1.4.3. Read run status

While the measurement is running on the DPP, most users want to get status information regarding the run. Typically, the minimum needed information is whether the run is still active or not. If only this information is required the parameter “Run Status” (ID 5) can be used. Since “Run Status” (ID 5) uses the standard response syntax, a 4-byte data frame is required:

5	0	0	0

Since “Run Status” (ID 5) uses the standard response syntax, a 4-byte data frame will be received from the device:

5	0	0	X

The second byte indicates the status 0, which means the request was processed successfully. X represents the run status acquired from the device. X=1 means the run is still active, while X=0 indicates that no run is active (e.g., stop condition has been met or stop run command was received).

In case the user wants to read more status information (e.g., count rates or current realtime) the parameter “Run Statistics” (ID 18) should be used instead. With this parameter not only the run status but also further statistics can be read. Since “Run Statistics” (ID 18) uses the standard request syntax, a 4-byte data frame is needed:

18	0	0	0

The response for “Runtime Statistics Read” (ID 18) has a special syntax described above (13 x 4 byte) with the following structure:

5	0	0	X
6	0	MSByte of “Realtime Low”	LSByte of “Realtime Low”
7	0	MSByte of “Run Realtime High”	LSByte of “Run Realtime High”
8	0	MSByte of “Run Livetime Low”	LSByte of “Run Livetime Low”
9	0	MSByte of “Run Livetime High”	LSByte of “Run Livetime High”
10	0	MSByte of “Run Output Counts Low”	LSByte of “Run Output Counts Low”
11	0	MSByte of “Run Output Counts High”	LSByte of “Run Output Counts High”
12	0	MSByte of “Run Input Counts Low”	LSByte of “Run Input Counts Low”
13	0	MSByte of “Run Input Counts High”	LSByte of “Run Input Counts High”
14	0	MSByte of “Run Output Count Rate Low”	LSByte of “Run Output Count Rate Low”
15	0	MSByte of “Run Output Count Rate High”	LSByte of “Run Output Count Rate High”
16	0	MSByte of “Run Input Count Rate Low”	LSByte of “Run Input Count Rate Low”
17	0	MSByte in „Output Count Rate High”	LSByte in „Output Count Rate High”

X represents the run status acquired from the device. X=1 means the run is still active, while X=0 indicates that no run is active (e.g., stop condition has been met or stop run command was received).

1.4.4. Read MCA data:

After completion of the run or during a running acquisition MCA data can be read out. For this purpose, the parameter “MCA read” (ID 19) is used. The request equals standard syntax with a 4-byte data frame:

19	0	0	0

The parameter “MCA Read” (ID 19) has a special syntax described above consisting just of the MCA data (no header). The returned data size is determined by the current settings of “MCA Number of Bins” (ID 20) and “MCA Bytes per Bin” (ID 21). The counts of each bin are transferred with the LSByte first. Transmission begins with bin 0.

Example: Parameter ID 20 „MCA Number of Bins” is set to 13 and parameter ID 21 „MCA Bytes per Bin” is set to 2. “MCA Read” returns 2^13 x 2 byte = 8192 x 2 byte = 16384 byte. First byte of the data is the LSByte of Bin 0 and the last transferred byte is the MSByte of bin 8191:

LSByte of bin 0	MSByte of bin 0	LSByte of bin 1	MSByte of bin 1	…	LSByte of bin 8191	MSByte of bin 8191

2. DPP3 user data concept

In the following the internal management of parameter values in the DPP3 will be described.

2.1. Parameter values in the working copy

The DPP3 is configured by the values of parameters. This applies not only to the pulse processing functionality (e.g., peaking times) but also to system information data (e.g., firmware version or board temperature), communication settings (e.g., network IP address) or process control (e.g., start a run). A unique number ("Parameter ID") between 0 and 255 is assigned to each parameter of the DPP3. Currently not all of the 256 possible IDs are used. For each parameter ID a 2-byte memory area is reserved in the FPGA of the DPP3. The parameter memory in the FPGA therefore has a size of 256 x 2 byte. Since all internal functions of the DPP (e.g., calculation of digital filters, network interface, …) use this parameter memory is referred to as working copy in the following. Parameter values of the working copy can be read out and manipulated using the communication interfaces (see: 1. DPP3 command set).

2.2. Non-volatile parameter values

Since the working copy is stored in volatile memory inside the FPGA, the settings are lost at every power cycle. In order to save settings an additional non-volatile memory is included in the DPP3. This memory unit is referred to as "user data memory". The user data memory can exchange data with the FPGA over a serial interface on the DPP3.

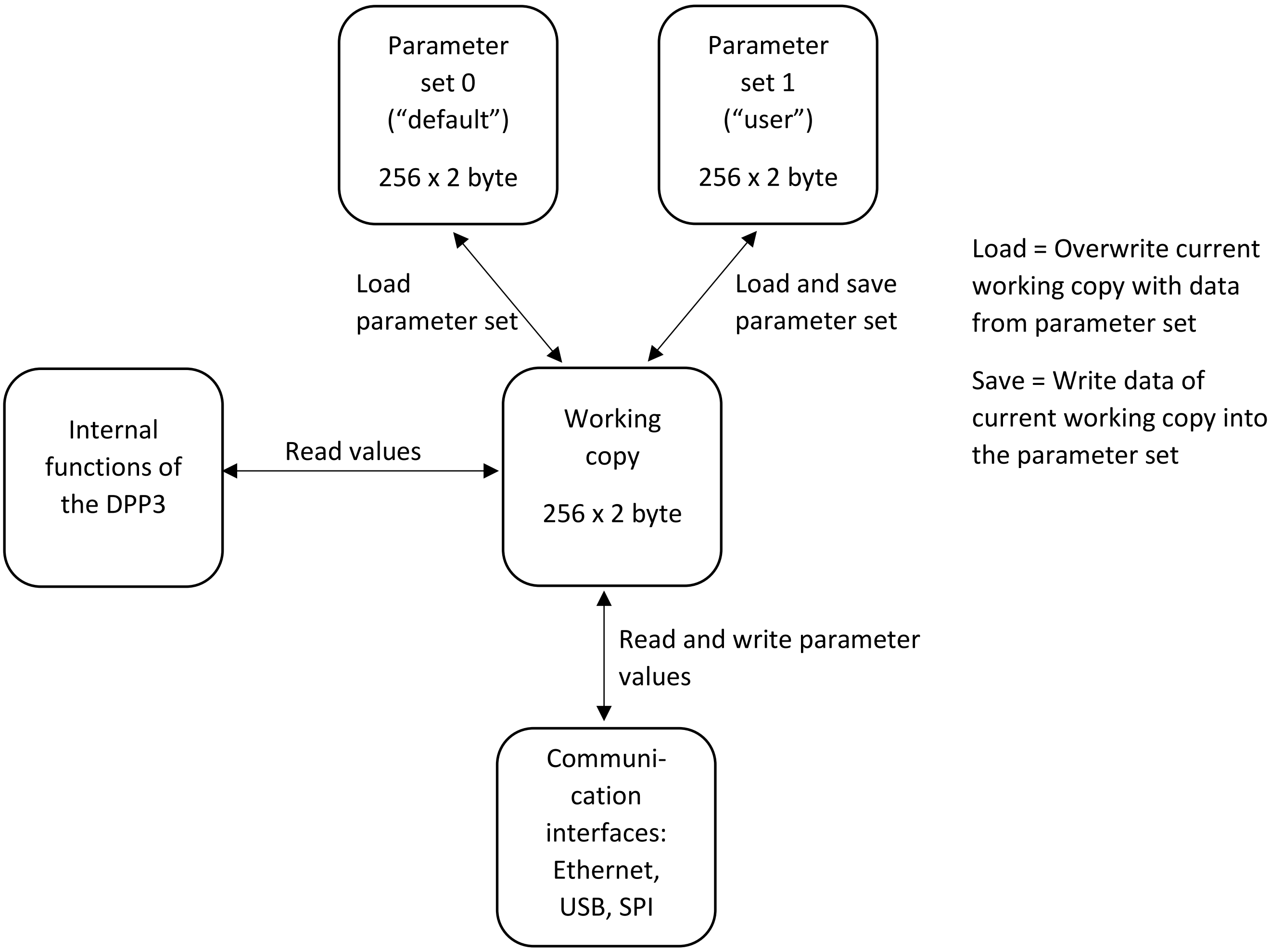
The user data memory contains two complete sets of parameters (each 256 x 2 byte):

- Parameter set 0 is referred to as default parameter set. This parameter set will be written by KETEK during the production process. The user can load this parameter set but he cannot overwrite the set. The default parameter set is meant to be a recovery option in case a faulty parameter set was defined by the user.
- Parameter set 1 is referred to as user parameter set. This parameter set will be written by KETEK during the production process equally to the default parameter set. The user can load this parameter set and is also able to override saved values. The user parameter set is designed to allow the user to store preferred settings non-volatile.

The user is not able to directly access the user data memory. Setting parameter values using the command set only effects the working copy. For utilization of the user data memory the parameters "Parameter Set Load" (ID 64) and "Parameter Set Save" (ID 65) are introduced. These parameters have the following functions:

- "Parameter Set Load" (ID 64): The current working copy is overwritten by the parameter values stored in the loaded parameter set. When requesting "Parameter Set Load" (ID 64) the data bytes can either be 0 in order to load the default parameter set into the working copy or 1 in order to load the user parameter set into the working copy.
- "Parameter Set Save" (ID 64): The current working copy is stored in the parameter set. When requesting "Parameter Set Save" (ID 64) the data bytes have to be 1 since only the user parameter set of the user data memory can be overwritten. Requesting "Parameter Set Save" (ID 64) with data bytes of 0 will lead to an error code.

The following diagram schematically shows the organization of parameters described above:



When the DPP3 is powered on the user parameter set will be loaded into the working copy. In this way the last settings saved by the user are available after each power cycle. Changing parameter values using the communication interfaces only affects the working copy. The changes will be applied to the internal functions of the DPP3 immediately though. However, as long as "Parameter Set Save" (ID 64) is not called, the changes are only saved volatile and will be lost after the next power cycle. If the user wishes to keep the changes "Parameter Set Save" (ID 64) has to be called. In this case the parameter set will be preserved after a power cycle.

2.3. Recovery of the DPP3 using the default-button

In case the user saved a faulty or unknown parameter set into the user parameter set, the default-button of the interface board can be used for recovery. In order to activate the recovery, the user has to switch off the power of the DPP3 first. Then push the recovery button while powering the DPP3 and keep that button pushed at least for another 3s. During the recovery process the user parameter set (1) is overwritten with the values of the default parameter set (0). Therefore, after the recovery the user parameter set (1) is identical with the default parameter set (0) and the user can continue with default settings.

As an alternative for pushing the button on the interface board provided by KETEK, the user can pull the pin 37 "FPGA_DFLT/MCU_REQFBL" of the VICO-DV 3.0 connector to high (3.3V). This is useful in case a custom interface board is used.

3. Update of the DPP firmware

The DPP3 provides the opportunity to update the DPP firmware using any supported communication interface (USB, Ethernet, SPI). This feature will be described in the following.

3.1. Firmware memory concept

The firmware memory of the DPP3 is divided into two separate areas. Each memory area is capable of saving one firmware file. The firmware file in the first area will be called “golden image” while the firmware file in the second area will be called “update image”:

- The golden image is stored into the firmware memory during the production process at KETEK using a programming interface. The latest DPP firmware version at the time of production will be used as golden image. The golden image cannot be modified using communication interfaces and can only be changed by KETEK using the programming interface.
- No update image will be stored during the production process at KETEK. Therefore, this memory area is empty in delivery state. Using the communication interfaces firmware data can be written into or read out of this memory area. Also, the update image can be deleted using the DPP3 command set.

When the DPP3 is powered on the first step is trying to boot the update image. Therefore, the update image will be loaded into the FPGA and boot process is started. In case the boot fails (e.g., no update image is stored, update image was corrupted) the DPP3 will fall back to the golden image. In this case the golden image will now be loaded into the FPGA and the boot process is started again. Since the golden image cannot be modified in field, this firmware will always be bootable.

Using this firmware memory concept, it is assured that the DPP3 still can be used after a failed update of the firmware (e.g., power or communication loss during update). In this case the golden image is booted and the user can repeat the firmware update.

3.2. Tools for updating the firmware

In order to update the firmware of the DPP3 via USB or via Ethernet KETEK provides the software tool “VICOUpdate”. VICOUpdate can be controlled using a graphical user interface or a command-line shell. With VICOUpdate a safe firmware update can be done with few clicks or commands. VICOUpdate is based on the API “VICOLib”. Further information can be found in the corresponding documentation.

If the firmware update process should be integrated in a custom application using VICOLib the API functions “deleteFirmware()”, “readFirmwareSection()” and “writeFirmwareSection()” can be used. More information on how to build a safe update process using these functions are available on request.

Furthermore, updating the firmware using low-level communication (USB/Ethernet without VICOLib or in case of SPI communication) is supported. This process will be described in the following.

3.3. Updating the firmware using low-level communication

In the following the process for updating the DPP firmware using low-level communication will be described. Information regarding the used parameters can also be found in the section 1 DPP3 command set. The necessary steps are:

3.3.1. Open the firmware file

The firmware file will be delivered as hex-file and the firmware version is noted in the file name (e.g., “esw-xv3.0-fpga-0.3.2.0.hex”) for version 0.3.2.0. The file contains the firmware data in ASCII-coded format and can be opened using common text editors. Within the hex-file 2192012 bytes of data can be found. However, the area in the firmware memory has a size of 4194304 bytes due to the given segmentation of the memory device and due possibly larger firmware file in future products. Therefore, the host software has to append 2002292 bytes (4194304 bytes - 2192012 bytes) of data to the firmware data read from the hex-file. The appended data should be all 0xFF. In this way the final firmware data with a length of 4194304 bytes are generated: The first 2192012 bytes contain the content of the hex-file and the last 2002292 bytes are all 0xFF. This final file can be transmitted to the device in the following.

3.3.2. Segmentation of the firmware

The final firmware data with a length of 4194304 bytes has to be divided into 4096 sections with a length of 1024 bytes each. Therefore, the host software has to extract blocks with 1024 bytes from the firmware data successively. The first block is referred to as section number 0 and the last block will be noted as section 4095.

3.3.3. Unlock update feature on the DPP3

In order to prevent the user from unintended changes on the firmware by sending certain requests by accident, the relevant parameters have to be unlocked first. To gain access to the firmware update feature the user first has to write a value of 18007 (0x4657) to the parameter "Service Code Low" (ID 94) and a value of 21840 (0x5550) to the parameter "Service Code High" (ID 95). Both parameters use the standard request syntax and the low-level data frames therefore are:

Parameter ID 94	**Command: Write**	**MSB of 18007**	**LSB of 18007**
94 (= 0x5E)	0x01	0x46	0x57

and

Parameter ID 95	**Command: Write**	**MSB of 21840**	**LSB of 21840**
95 (= 0x5F)	0x01	0x55	0x50

Both parameters use the standard response syntax and the received low-level data frames therefore are:

Parameter ID 94	**Status: Success**	**MSB of 18007**	**LSB of 18007**
94 (= 0x5E)	0x00	0x46	0x57

and

Parameter-ID 95	**Status: Success**	**MSB of 21840**	**LSB of 21840**
95 (= 0x5F)	0x00	0x55	0x50

Afterwards access to the parameters "Delete Firmware" (ID 91), "Write Firmware Section" (ID 92), and "Read Firmware Section" (ID 93) is granted. The access is lost when the device is power cycled or if any other data are written to "Service Code Low" (ID 94) and "Service Code High" (ID 95).

3.3.4. Delete the current update image on the DPP3

Before a new firmware can be transferred to the DPP the current update image in the firmware memory of the DPP3 has to be deleted. This step is mandatory since only erased parts of the memory device can be written. In order to delete the update image on the DPP3 the parameter "Delete

Firmware" (ID 91) has to be called. The parameter uses standard request syntax and the low-level data frame to delete the firmware therefore is:

Parameter ID 91	Command: Irrelevant	MSB: Irrelevant	LSB: Irrelevant
91 (= 0x5B)	0x00	0x00	0x00

The DPP3 then erases the update image in firmware memory. After completion of this process (typically 30s, maximum 90s) the response in standard syntax will be returned:

Parameter-ID 91	Status: Success	MSB: Irrelevant	LSB: Irrelevant
91 (= 0x5B)	0x00	0x00	0x00

This indicates that the update image memory area was erased successfully.

3.3.5. Write firmware sections to the DPP3 and verify data

After the memory area of update image has been erased the new firmware can be transferred to the DPP3. In order to ensure a safe update process, we recommend to read back and verify each firmware section after it has been written.

The transfer of the firmware must to be started at the highest section number 4095. Trying to write any other section number than 4095 after the firmware has been deleted will result in an error code 0x02 (see: 1.3.1 Standard structure of responses) In order to transfer a firmware section, the parameter "Write Firmware Section" (ID 92) is used. This parameter has a special request syntax (see: 1.2.2 Special structure of requests) with a length of 1028 bytes. The low-level data frame used to write the section 4095 looks like this:

Parameter ID 92	Command: Irrelevant	Section number: MSB of 4095	Section number: LSB of 4095	1st data byte of firmware section 4095	…	1024th data byte of firmware section 4095
92 (= 0x5C)	0x01	0x0F	0xFF	??	…	??

The DPP3 transfers the 1024 bytes of firmware data into the specified section 4095 of the firmware memory. After completion of the write process (typically 1ms) the response in standard syntax will be returned:

Parameter ID 92	Status: Success	Section number: MSB of 4095	Section number: LSB of 4095
92 (= 0x5C)	0x00	0x0F	0xFF

As a next step the written data should be verified. Therefore, the just written firmware section 4095 is read out using the parameter "Read Firmware Section" (ID 93). The parameter uses standard request syntax and the low-level data frame to delete the firmware therefore is:

Parameter ID 93	Command: Irrelevant	Section number: MSB of 4095	Section number: LSB of 4095
93 (= 0x5B)	0x00	0x0F	0xFF

The DPP3 now reads out the data from the specified section 4095 of the firmware memory. The response has a special syntax with a length of 1028 bytes (see: 1.3.2 Special structure of responses) using the following structure:

Parameter ID 93	**Status: Success**	**Section number: MSB of 4095**	**Section number: LSB of 4095**	**1st data byte of firmware section 4095**	**…**	**1024th data byte of firmware section 4095**
93 (= 0x5B)	0x00	0x0F	0xFF	??	…	??

Now the user should compare the 1024 bytes of firmware data read from the DPP3 with the previously transferred 1024 bytes of firmware data in order to ensure correct data transmission. In case all data bytes are equal, the user can continue to write and verify the next smaller section number (here section number 4094). The process is continued until section number 0 has been written and verified by the user. Afterwards the firmware file has been written into the firmware memory.

In case the verification step fails at any section number (read data are not equal to previously written data), the user should stop the update process and erase the firmware on the DPP3 since it might be corrupted. It is not possible to write just the corrupted firmware section again, since only erased memory parts can be written.

3.3.6. Reboot of the DPP3

So far in the update process the firmware file has been transferred into the firmware memory of the DPP3. This however does not affect the firmware currently loaded in the FPGA. This firmware was booted when the DPP3 was powered and will not be modified during the update process. In order to use the previously transferred firmware, a power cycle of the DPP3 is necessary. Afterwards the boot process described in Firmware memory concept will be started and the DPP3 first tries to boot the update image. If this is successful the previously transferred firmware will be used and will also be booted after following power cycles. In case the update image is not bootable, the DPP3 falls back to the golden image and the user can repeat the update process. In order to ensure that the update image was booted successfully we recommend to read out the firmware version after the power cycle using the parameter IDs 66 to 69 and compare the firmware version to the one noted in the hex-filename.