NT-AN1237-4.2

Application Note

10G-KR Software API Guide

Released

September 2017

Contents

Tables

Figures

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

Revision 1.0

Revision 1.0 was published in September 2017. It was the first publication of this document.

2 Introduction

This document describes the software API usages for the Ethernet backplane functionality. The backplane functionality consists of:

- 802.3ap Clause 72 Link Training
- 802.3ap Clause 73 Auto negotiation
- 10GBase-KR FEC

The Auto negotiation function allows the link partners to negotiate the modes of operation—speed and FEC mode. This process is performed using the Differential Manchester Coding (DME) pages.

For JR2 Rev-C, only 10GBase-KR speed is supported and advertised.

Link training is performed as part of the aneg process. The aim is to establish optimal transmitter equalization settings for both Local Device (LD) and Link Partner (LP). The training state machine is implemented in hardware and does not require software intervention. The results are mapped to the output buffer settings automatically.

FEC capabilities are advertised for the 10Gbase-KR interface-FEC ability and FEC enable request. FEC is enabled on the link only if both the link partners advertise that they have FEC ability and at least one of them requests to enable FEC. The enable/disable operation is performed during a status-get call from the application based upon the aneg results.

The optimal receiver settings need to be chosen based on the media and it is applied when the port is configured. The port serdes configuration offers few different offsets, including, VTSS_SD10G_MEDIA_10G_KR.

The software API supports the configuration of the above and status-get function. The status-get function must be integrated into a thread and polled, for example, at 100 ms interval.

2.1 Audience

This document is for software and/or application developers who need to use or implement the Backplane SW-API in their application.

3 Backplane SW-API

The following sections discusses the backplane software API.

3.1 Configure the port speed and Serdes mode

The following is a configuration example of port setup and serdes mode.

```
vtss_port_conf_set   (const vtss_inst_t inst,
                      const vtss_port_no_t port_no,
                      const vtss_port_conf_t *const conf);


typedef struct
{
vtss_port_interface_t if_type; /**< Interface type */
vtss_port_speed_t speed; /**< Port speed */
. . . . .
. . . . .
vtss_port_serdes_conf_t serdes; /**< Serdes settings (for SFI interface) */
} vtss_port_conf_t;
```

When configuring the SFI 10G interface, set the parameters as following:

vtss_port_conf_t ::if_type = VTSS_PORT_INTERFACE_SFI

vtss_port_conf_t ::speed = VTSS_SPEED_10G

vtss_port_conf_t ::serdes. ::media_type = vtss_sd10g_media_type_t

The vtss_sd10g_media_type_t can be any of the following:

```
typedef enum {
VTSS_SD10G_MEDIA_SR, /**< Short Range > */
VTSS_SD10G_MEDIA_ZR, /**< ZR with APC hardware algorithm > */
VTSS_SD10G_MEDIA_DAC, /**< DAC with APC hardware algorithm > */
VTSS_SD10G_MEDIA_BP, /**< Backplane > */
VTSS_SD10G_MEDIA_B2B, /**< Bord to Board > */
VTSS_SD10G_MEDIA_10G_KR, /**< 10G Base KR > */
VTSS_SD10G_MEDIA_PR_NONE /**< No preset > */
}
vtss_sd10g_media_type_t;
```

This depends on the media interface, but for a backplane the VTSS_SD10G_MEDIA_10G_KR should be used. This parameter controls the preset of the serdes receiver.

3.2 Auto-Negotiation and Training Configuration

The following is a configuration example of auto-negotiation and training.

```
vtss_port_10g_kr_conf_set    (const vtss_inst_t inst,
                              const vtss_port_no_t port_no,
                              const vtss_port_10g_kr_conf_t *const conf);


typedef struct {
vtss_port_10g_kr_aneg_t aneg; /**< 10G-KR Aneg apability, 802.3ap Clause 73 */
vtss_port_10g_kr_train_t train; /**< 10G-KR Training parameters, 802.3ap Clause 72 */
} vtss_port_10g_kr_conf_t;


typedef struct {
BOOL enable; /**< 10G KR Autoneg enable */
BOOL adv_10g; /**< Advertise 10G */
BOOL fec_abil; /**< Set FEC ability */
BOOL fec_req; /**< Set FEC request */
} vtss_port_10g_kr_aneg_t;


typedef struct {
BOOL enable; /**< Enable 10G KR training,
BER method used */} vtss_port_10g_kr_train_t;
```

The vtss_port_10g_kr_conf_set() function enables and starts auto-negotiation and training. The ability is configured as described. Training can only be enabled or disabled.

3.3 Getting Auto-Negotiation and Ttraining Results

The outcome of the auto-negotiation and training is fetched through status-get function. It is necessary to poll this function to keep the aneg and training consistent to monitor if aneg restarts due to link down, or if partner restarts aneg.

```
vtss_rc vtss_port_10g_kr_status_get  (const vtss_inst_t inst,
                                      const vtss_port_no_t port_no,
                                      vtss_port_10g_kr_status_t *const status);


typedef struct {
vtss_port_10g_kr_status_aneg_t aneg; /**< Aneg structure */
vtss_port_10g_kr_status_train_t train; /**< Training structure */
vtss_port_10g_kr_status_fec_t fec; /**< FEC structure */}
vtss_port_10g_kr_status_t;


BOOL complete; /**< Aneg completed successfully */
BOOL active; /**< Aneg is running between LD and LP */
BOOL request_10g; /**< 10G rate is negotiated (needs to be configured) */
BOOL request_1g; /**< 1G rate is negotiated (needs to be configured) */
BOOL request_fec_change; /**< FEC enable is negotiated (needs to be enabled) */
BOOL fec_enable; /**< FEC disable is negotiated (needs to be disabled) */
u32 sm; /**< (debug) Aneg state machine */
BOOL lp_aneg_able; /**< (debug) Link partner aneg ability */
BOOL block_lock; /**< (debug) PCS block lock */
} vtss_port_10g_kr_status_aneg_t;
```

Table 1 • vtss_rc vtss_port_10g_kr_status_get

BOOL complete	Specifies that the aneg process finishes successfully and the results can be viewed through the 'request' Booleans.
BOOL active	Specifies that the aneg process is still ongoing.

BOOL request_10g	Specifies that the aneg process results in 10G speed. If the speed is already 10G, then nothing needs to be done.
BOOL request_1g	Specifies that the aneg process results in 1G speed. Note that, 1000 Base-KX is not supported in JR2-RevC, that is, this flag should never be set.
BOOL request_fec_change	Specifies that the aneg process results in FEC state change request, enable or disable.
BOOL fec_enable	If the request_fec_change is set, then the FEC state should be enabled ('fec_enable' = 1) or disabled ('fec_enable' = 0).
u32 sm	Specifies the state of the aneg state machine (for debug purposes).
BOOL lp_aneg_able	Specifies the aneg state of the link partners—whether the LP aneg is enabled?
BOOL block_lock	Specifies whether the PCS block locks on this port. False means no link.

```
typedef struct {
BOOL complete; /**< Training completed successfully */
u8 cm_ob_tap_result; /**< The minus 1 coefficient c(-1). Range: -32..31 */
u8 cp_ob_tap_result; /**< The 0 coefficient c(0). Range: -32..31 */
u8 c0_ob_tap_result; /**< The plus 1 coefficient c(1). Range: -32..31 */
} vtss_port_10g_kr_status_train_t;
```

Table 2 • vtss_port_10g_kr_status_train_t

BOOL complete	Specifies that the Training completed successfully.
u8 cm_ob_tap_result;	The minus 1 coefficient c(-1). 7-bit signed. Range: -32–31 Results of the training, for viewing only. This value is automatically transferred to the Serdes SD1 0G65_OB_CFG2_D_FILTER.
u8 cp_ob_tap_result;	The 0 coefficient c(0). 7-bit signed. Range: -32–31. Results of the training, for viewing only. This value is automatically transferred to the Serdes SD1 0G65_OB_CFG2_D_FILTER.
u8 c0_ob_tap_result;	The plus 1 coefficient c(1). 7-bit signed. Range: -32–31 Results of the training, for viewing only. This value is automatically transferred to the Serdes SD10G65_OB_CFG2_D_FILTER.

```
typedef struct {
BOOL enable; /**< FEC Enabled */
u32 corrected_block_cnt; /**< Corrected block count */
u32 uncorrected_block_cnt; /**< Un-corrected block count */
} vtss_port_10g_kr_status_fec_t;
```

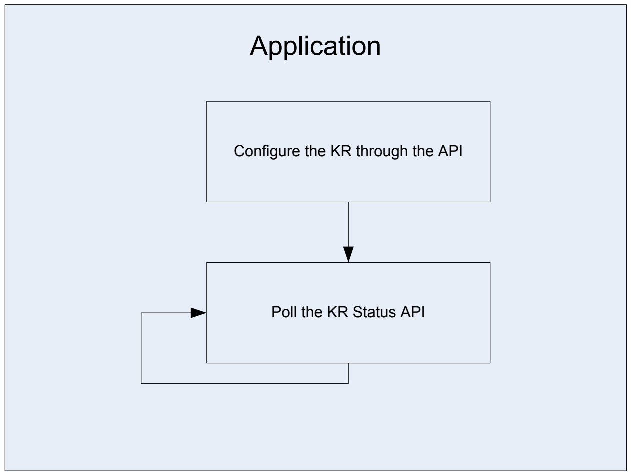
Table 3 • vtss_port_10g_kr_status_fec_t

BOOL enable	Specifies that the FEC is enabled.
u32 corrected_block_cnt ;	Specifies the corrected block counts.
u32 uncorrected_block_cnt	Specifies the un-corrected block counts.

4 Backplane SW-Application

The application controls the backplane configuration through the API and the following status polling.

Figure 1 • Application



Aneg is restarted during the restart of aneg from the link partner or if the cable is removed/inserted. The aneg process may take up to 500 ms. If the aneg does not complete during this period, then the aneg process is restarted by LD and/or LP. A poll time in every 100 ms should be enough to avoid aneg restarting.

4.1 Example Application

After the configuration is completed, the following thread is awoken:

```
static void kr_thread(vtss_addrword_t data)
{
mesa_port_10g_kr_status_t status;
mesa_port_10g_kr_conf_t conf;
mesa_port_no_t port_no;
u32 port_cnt = mesa_port_cnt(NULL);
for (;;) {
VTSS_OS_MSLEEP(misc.kr_thread_ms_poll);
vtss_flag_wait(&misc.flags, MISC_FLAG_PORT_FLAP, VTSS_FLAG_WAITMODE_OR);
for (port_no = 0; port_no < port_cnt; port_no++) {
if ((mesa_port_10g_kr_conf_get(NULL, port_no, &conf) != VTSS_RC_OK) || !conf.aneg.enable)
 {
continue;
}
// 10G KR surveillance
(void)(mesa_port_10g_kr_status_get(NULL, port_no, &status));
}
}
}
```

5 JR2-24 Reference Board Demo Application

10G-KR configuration and status can be performed from the interface mode.

The commands are "debug" commands and they are not saved anywhere.

5.1 10G-KR Configuration

For example, debug kr-aneg all starts aneg with everything advertised, starts training, and when finished a thread that polls the API is started.

"debug kr-aneg" shows the current configuration.

```
(config-if)# debug kr-aneg ?
adv-10g Advertise 10g
adv-1g Advertise 1G
all Advertise all
aneg-off Aneg disable
aneg-only Aneg only (no training)
fec-abil Advertise FEC ability
fec-req Request FEC ability
train-only Training only (no aneg)
```

5.2 10G-KR Status

debug kr-status shows the status. Note that the "request" Booleans are cleared after reading.

The FEC is enabled/disabled automatically through a "status_get" call based on the aneg results.

No further actions are needed from the user. Also note that the "poll thread" calls this function to be able to apply aneg results and restart aneg if needed.

```
(config-if)# debug kr-status
Port 25
LP aneg ability :Yes
Aneg active (running) :No
PCS block lock :Yes
Aneg complete :Yes
Enable 10G request :No
Enable 1G request :No
FEC change request :No
Training complete (BER method) :Yes
CM OB tap (7-bit signed) :-3 (125)
CP OB tap (7-bit signed) :-2 (126)
C0 OB tap (7-bit signed) :+23 (23)
FEC :Enabled
Corrected block count :0
Un-corrected block count :0
```

The above results show a 10G-KR link in an "Good" state.

5.3 10G-KR Polling

The polling thread can be controlled through the following debug command. The thread is also started through the "debug kr-aneg" command.

```
config-if)# debug kr-poll ?
disable Disable polling
enable Enable polling
poll Set poll time
```

The polling can be enabled/disabled and the interval can be configured (default 100 ms)

5.4 10G-KR API Trace

Trace is valuable for debugging. Enable trace from the root with the following:

```
# deb trace module level api_cil port info
```

5.5 10G-KR Register Dump

Dump KR-Aneg registers:

```
# deb sym read XGKR1[0]
Register                          Value      Decimal    31      24 23      16 15       8 7
       0
XGKR1[0]:KR_7X0000:KR_7X0000      0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:KR_7X0001:KR_7X0001      0x00000008          8
0000.0000.0000.0000.0000.0000.0000.1000
XGKR1[0]:LD_ADV:KR_7X0010         0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LD_ADV:KR_7X0011         0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LD_ADV:KR_7X0012         0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_BASE_PAGE_0:KR_7X0013 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_BASE_PAGE_1:KR_7X0014 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_BASE_PAGE_2:KR_7X0015 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LD_NEXT_PAGE:KR_7X0016   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LD_NEXT_PAGE:KR_7X0017   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LD_NEXT_PAGE:KR_7X0018   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_NEXT_PAGE:KR_7X0019   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_NEXT_PAGE:KR_7X001A   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_NEXT_PAGE:KR_7X001B   0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:KR_7X0030:KR_7X0030      0x00000003          3
0000.0000.0000.0000.0000.0000.0000.0011
XGKR1[0]:AN_CFG0:AN_CFG0          0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:BL_TMR:BL_LSW            0x0000d6af      54959
0000.0000.0000.0000.1101.0110.1010.1111
XGKR1[0]:BL_TMR:BL_MSW            0x00000029         41
0000.0000.0000.0000.0000.0000.0010.1001
XGKR1[0]:AW_TMR:AW_LSW            0x0000c3df      50143
0000.0000.0000.0000.1100.0011.1101.1111
XGKR1[0]:AW_TMR:AW_MSW            0x00000016         22
0000.0000.0000.0000.0000.0000.0001.0110
XGKR1[0]:LFLONG_TMR:LFLONG_LSW    0x000000e2        226
0000.0000.0000.0000.0000.0000.1110.0010
XGKR1[0]:LFLONG_TMR:LFLONG_MSW    0x0000012d        301
0000.0000.0000.0000.0000.0001.0010.1101
XGKR1[0]:LFSHORT_TMR:LFSHORT_LSW  0x0000aff4      45044
0000.0000.0000.0000.1010.1111.1111.0100
XGKR1[0]:LFSHORT_TMR:LFSHORT_MSW  0x0000001b         27
0000.0000.0000.0000.0000.0000.0001.1011
XGKR1[0]:LP_TMR:LP_LSW            0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:LP_TMR:LP_MSW            0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:TR_TMR:TR_LSW            0x000000e2        226
0000.0000.0000.0000.0000.0000.1110.0010
XGKR1[0]:TR_TMR:TR_MSW            0x0000012d        301
0000.0000.0000.0000.0000.0001.0010.1101
XGKR1[0]:PD_TMR:PD_LSW            0x000000e2        226
0000.0000.0000.0000.0000.0000.1110.0010
XGKR1[0]:PD_TMR:PD_MSW            0x0000012d        301
0000.0000.0000.0000.0000.0001.0010.1101
XGKR1[0]:KR10G_TMR:KR10G_LSW      0x00001a80       6784
0000.0000.0000.0000.0001.1010.1000.0000
XGKR1[0]:KR10G_TMR:KR10G_MSW      0x00000006          6
0000.0000.0000.0000.0000.0000.0000.0110
XGKR1[0]:KR3G_TMR:KR3G_LSW        0x00001a80       6784
0000.0000.0000.0000.0001.1010.1000.0000
XGKR1[0]:KR3G_TMR:KR3G_MSW        0x00000006          6
0000.0000.0000.0000.0000.0000.0000.0110
XGKR1[0]:KR1G_TMR:KR1G_LSW        0x00001a80       6784
0000.0000.0000.0000.0001.1010.1000.0000
XGKR1[0]:KR1G_TMR:KR1G_MSW        0x00000006          6
0000.0000.0000.0000.0000.0000.0000.0110
XGKR1[0]:AN_HIST:AN_HIST          0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
```

```
XGKR1[0]:AN_SM:AN_SM              0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR1[0]:AN_STS0:AN_STS0          0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000

Dump KR-Training registers:
# deb sym read XGKR0[0]
Register                           Value      Decimal    31      24 23      16 15       8
7        0
XGKR0[0]:KR_1X0096:KR_1X0096       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:KR_1X0097:KR_1X0097       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:KR_1X0098:KR_1X0098       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:KR_1X0099:KR_1X0099       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:KR_1X009A:KR_1X009A       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:KR_1X009B:KR_1X009B       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_CFG0:TR_CFG0           0x0000401b      16411
0000.0000.0000.0000.0100.0000.0001.1011
XGKR0[0]:TR_CFG1:TR_CFG1           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_CFG2:TR_CFG2           0x000007c1       1985
0000.0000.0000.0000.0000.0111.1100.0001
XGKR0[0]:TR_CFG3:TR_CFG3           0x00000034         52
0000.0000.0000.0000.0000.0000.0011.0100
XGKR0[0]:TR_CFG4:TR_CFG4           0x000007d1       2001
0000.0000.0000.0000.0000.0111.1101.0001
XGKR0[0]:TR_CFG5:TR_CFG5           0x0000003a         58
0000.0000.0000.0000.0000.0000.0011.1010
XGKR0[0]:TR_CFG6:TR_CFG6           0x00000e14       3604
0000.0000.0000.0000.0000.1110.0001.0100
XGKR0[0]:TR_CFG7:TR_CFG7           0x00000f80       3968
0000.0000.0000.0000.0000.1111.1000.0000
XGKR0[0]:TR_CFG8:TR_CFG8           0x00000055         85
0000.0000.0000.0000.0000.0000.0101.0101
XGKR0[0]:TR_CFG9:TR_CFG9           0x00000014         20
0000.0000.0000.0000.0000.0000.0001.0100
XGKR0[0]:TR_GAIN:TR_GAIN           0x0000a000      40960
0000.0000.0000.0000.1010.0000.0000.0000
XGKR0[0]:TR_COEF_OVRD:TR_COEF_OVRD 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_STAT_OVRD:TR_STAT_OVRD 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_OVRD:TR_OVRD           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_STEP:TR_STEP           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_MTHD:TR_MTHD           0x00000918       2328
0000.0000.0000.0000.0000.1001.0001.1000
XGKR0[0]:TR_BER_THR:TR_BER_THR     0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_BER_OFS:TR_BER_OFS     0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_LUTSEL:TR_LUTSEL       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_BRKMASK:BRKMASK_LSW    0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_BRKMASK:BRKMASK_MSW    0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:OBCFG_ADDR:OBCFG_ADDR     0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:APC_TMR:APC_TMR           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:WT_TMR:WT_TMR             0x00000a08       2568
0000.0000.0000.0000.0000.1010.0000.1000
XGKR0[0]:MW_TMR:MW_TMR_LSW         0x0000a30a      41738
0000.0000.0000.0000.1010.0011.0000.1010
XGKR0[0]:MW_TMR:MW_TMR_MSW         0x00000133        307
0000.0000.0000.0000.0000.0001.0011.0011
XGKR0[0]:TR_STS1:TR_STS1           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_STS2:TR_STS2           0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
```

```
XGKR0[0]:TR_TAPVAL:TR_CMVAL        0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_TAPVAL:TR_C0VAL        0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_TAPVAL:TR_CPVAL        0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_FRAMES_SENT:FRSENT_LSW 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_FRAMES_SENT:FRSENT_MSW 0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_LUT:LUT_LSW            0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_LUT:LUT_MSW            0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_ERRCNT:TR_ERRCNT       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_SM_HIST:HIST_LSW       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_SM_HIST:HIST_MSW       0x00000000          0
0000.0000.0000.0000.0000.0000.0000.0000
XGKR0[0]:TR_REC:TR_CM_LIM_REC      0x00004040      16448
0000.0000.0000.0000.0100.0000.0100.0000
XGKR0[0]:TR_REC:TR_CM_DME_REC      0x00000040         64
0000.0000.0000.0000.0000.0000.0100.0000
XGKR0[0]:TR_REC:TR_C0_LIM_REC      0x00004040      16448
0000.0000.0000.0000.0100.0000.0100.0000
XGKR0[0]:TR_REC:TR_C0_DME_REC      0x00000040         64
0000.0000.0000.0000.0000.0000.0100.0000
XGKR0[0]:TR_REC:TR_CP_LIM_REC      0x00004040      16448
0000.0000.0000.0000.0100.0000.0100.0000
XGKR0[0]:TR_REC:TR_CP_DME_REC      0x00000040         64
0000.0000.0000.0000.0000.0000.0100.0000

Dump XFI control :

# deb sym read XGXFI[0]:XFI_CONTROL
Register                                     Value
XGXFI[0]:XFI_CONTROL:KR_CONTROL              0x00000000
XGXFI[0]:XFI_CONTROL:XFI_MODE                0x00000016
XGXFI[0]:XFI_CONTROL:XFI_STATUS              0x01000000
XGXFI[0]:XFI_CONTROL:INT_CTRL                0xa8000000
XGXFI[0]:XFI_CONTROL:SSF_HYST_ENA_CTRL       0x00000000
XGXFI[0]:XFI_CONTROL:SSF_HYST_TIMING_CTRL    0x04c40017
XGXFI[0]:XFI_CONTROL:HSS_STICKY              0x000ff000
XGXFI[0]:XFI_CONTROL:HSS_MASK                0x00000000
XGXFI[0]:XFI_CONTROL:HSS_STATUS              0x00054000
XGXFI[0]:XFI_CONTROL:DATA_VALID_DETECT_CTRL 0x00000000
```

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

VPPD-04466