SECP

Created by: IDesignSpec

Table of Content

1.4.14	reg : SYS_ETS_DOORBELL[0]	0x00000000	0x000B0220, 0x000B0224...	65
1.4.14	reg : SYS_ETS_DOORBELL[1]	0x00000000	0x000B0220, 0x000B0224...	65
1.4.14	reg : SYS_ETS_DOORBELL[2]	0x00000000	0x000B0220, 0x000B0224...	65
1.4.15	reg : SYS_ROT_STATUS_LOCK	0x00000000	0x000B0240	65
1.4.16	reg : SYS_LOCAL_STATUS_LOCK	0x00000000	0x000B0244	65
1.4.17	reg : SYS_ROT_STATUS	0x00000000	0x000B0248	65
1.4.18	reg : SYS_LOCAL_STATUS	0x00000000	0x000B024C	65
1.4.19	reg : SYS_CLK_CTRL	0x00000000	0x000B0260	66
1.4.20	reg : SYS_RNG_CTRL	0x00000001	0x000B0264	66
1.4.21	reg : SYS_OTP_CTRL	0x00000000	0x000B0268	66
1.4.22	reg : SYS_TESTMUX_CTRL	0x00000000	0x000B026C	67
1.4.23	reg : SYS_CRYPTO_MEM_SEL	0x00000001	0x000B0270	67
1.4.24	reg : SYS_NMI_VECTOR	0x00000000	0x000B0274	67
1.4.25	reg : SYS_INTR_STATE	0x00000000	0x000B0280	67
1.4.26	reg : SYS_INTR_ENABLE	0x00000000	0x000B0284	69
1.4.27	reg : SYS_INTR_TEST	0x00000000	0x000B0288	69
1.4.28	reg : SYS_ECC_CTRL_SECP_RAM	0x00000000	0x000B0290	70
1.4.29	reg : SYS_ECC_STATUS_SECP_RAM1	0x00000000	0x000B0294	70
1.4.30	reg : SYS_ECC_DATA_SECP_RAM	0x00000000	0x000B0298	70
1.4.31	reg : SYS_ECC_ADDR_SECP_RAM	0x00F00000	0x000B029C	70
1.4.32	reg : SYS_ECC_CTRL_SECP_RAM	0x00000000	0x000B02A0	71
1.4.33	reg : SYS_ECC_STATUS_SECP_RAM2	0x00000000	0x000B02A4	72
1.4.34	reg : SYS_ECC_DATA_SECP_RAM	0x00000000	0x000B02A8	72
1.4.35	reg : SYS_ECC_ADDR_SECP_RAM	0x00F00000	0x000B02AC	72
1.4.36	reg : SYS_ECC_CTRL_KV_RAM	0x00000000	0x000B02B0	73
1.4.37	reg : SYS_ECC_STATUS_KV_RAM_TOP	0x00000000	0x000B02B4	73
1.4.38	reg : SYS_ECC_STATUS_KV_RAM	0x0000F000	0x000B02B8, 0x000B02BC...	74
1.4.38	reg : SYS_ECC_STATUS_KV_RAM[0]	0x0000F000	0x000B02B8, 0x000B02BC...	74
1.4.38	reg : SYS_ECC_STATUS_KV_RAM[1]	0x0000F000	0x000B02B8, 0x000B02BC...	74
1.4.38	reg : SYS_ECC_STATUS_KV_RAM[2]	0x0000F000	0x000B02B8, 0x000B02BC...	74
1.4.38	reg : SYS_ECC_STATUS_KV_RAM[3]	0x0000F000	0x000B02B8, 0x000B02BC...	74
1.4.39	reg : SYS_ECC_DATA_KV_RAM	0x00000000	0x000B02C8, 0x000B02CC...	75
1.4.39	reg : SYS_ECC_DATA_KV_RAM[0]	0x00000000	0x000B02C8, 0x000B02CC...	75
1.4.39	reg : SYS_ECC_DATA_KV_RAM[1]	0x00000000	0x000B02C8, 0x000B02CC...	75
1.4.39	reg : SYS_ECC_DATA_KV_RAM[2]	0x00000000	0x000B02C8, 0x000B02CC...	75

1.4.55	section : SYS_PM_TABLE[1]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[2]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[3]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[4]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[5]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[6]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[7]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[8]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[9]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[10]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[11]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[12]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[13]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[14]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[15]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[16]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[17]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[18]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[19]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[20]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[21]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[22]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[23]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[24]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[25]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[26]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[27]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[28]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[29]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[30]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55	section : SYS_PM_TABLE[31]		0x000B0400, 0x000B0408 ... 0x000B04FF	79
1.4.55.1	reg : CONTROL	0x00000000	0x000B0400, 0x0000000000...	79
1.4.55.2	reg : PAGE_ADDR	0x00000000	0x000B0404, 0x0000000000...	80
1.4.56	section : SYS_DMA_MPU		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[0]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[1]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[2]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[3]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[4]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[5]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[6]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56	section : SYS_DMA_MPU[7]		0x000B0500, 0x000B0510 ... 0x000B057F	80
1.4.56.1	reg : RANGE_START	0x00000000	0x000B0500, 0x0000000000...	80
1.4.56.2	reg : RANGE_END	0x00000FFF	0x000B0504, 0x0000000000...	80
1.4.56.3	reg : WRITE_PROTECT	0x00000000	0x000B0508, 0x0000000000...	81
1.4.56.4	reg : READ_PROTECT	0x00000000	0x000B050C, 0x0000000000...	81
1.4.57	reg : SYS_DMA_MPU_LOCK	0x00000000	0x000B0580	81
1.4.58	reg : SYS_EXTP_PERMIT	0x00000001	0x000B0600	81
1.4.59	reg : SYS_EXTP_PERMIT_LOCK	0x00000001	0x000B0604	82
1.4.60	reg : SYS_SCRATCH_2	0x00000000	0x000B0704	82
1.5	section : ROT_OTP		0x000C3000 - 0x000C3203	83
1.5.1	section : OTP_OTPC		0x000C3000 - 0x000C30FF	83
1.5.1.1	reg : OTP_OTPC_CONTROL	0x00000000	0x000C3000	83
1.5.1.2	reg : OTP_OTPC_ADDRESS	0x00000000	0x000C3004	83
1.5.1.3	reg : OTP_OTPC_CONTROL0	0x00000000	0x000C3008	83
1.5.1.4	reg : OTP_OTPC_STATUS0	0x0000200A	0x000C300C	83
1.5.1.5	reg : OTP_OTPC_STATUS1	0x00000000	0x000C3010	84
1.5.1.6	reg : OTP_OTPC_WRITE	0x00000000	0x000C3040, 0x000C3044...	85
1.5.1.6	reg : OTP_OTPC_WRITE[0]	0x00000000	0x000C3040, 0x000C3044...	85
1.5.1.6	reg : OTP_OTPC_WRITE[1]	0x00000000	0x000C3040, 0x000C3044...	85
1.5.1.7	reg : OTP_OTPC_READ	0x00000000	0x000C3080, 0x000C3084...	85
1.5.1.7	reg : OTP_OTPC_READ[0]	0x00000000	0x000C3080, 0x000C3084...	85
1.5.1.7	reg : OTP_OTPC_READ[1]	0x00000000	0x000C3080, 0x000C3084...	85
1.5.1.8	reg : OTP_OTPC_UNUSED	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[0]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[1]	0x00000000	0x000C3088, 0x000C308C...	85

1.5.1.8	reg : OTP_OTPC_UNUSED[2]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[3]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[4]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[5]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[6]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[7]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[8]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[9]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[10]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[11]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[12]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[13]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[14]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[15]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[16]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[17]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[18]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[19]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[20]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[21]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[22]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[23]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[24]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[25]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[26]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[27]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[28]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.1.8	reg : OTP_OTPC_UNUSED[29]	0x00000000	0x000C3088, 0x000C308C...	85
1.5.2	section : OTP_EFC		0x000C3100 - 0x000C31FF	85
1.5.2.1	reg : OTP_EFC_CONTROL	0x00000000	0x000C3100	85
1.5.2.2	reg : OTP_EFC_BYPASS_CONTRO	0x00050000	0x000C3104	86
1.5.2.3	reg : OTP_EFC_STATUS	0x00000000	0x000C3108	86
1.5.2.4	reg : OTP_EFC_POWER_CONTROl	0x00000002	0x000C310C	86
1.5.2.5	reg : OTP_EFC_ACCESS_TIMERS	0x00000000	0x000C3110	86
1.5.2.6	reg : OTP_EFC_ADDRESS	0x00000000	0x000C3114	86
1.5.2.7	reg : OTP_EFC_PGM_DATA	0x00000000	0x000C3118	87
1.5.2.8	reg : OTP_EFC_RD_DATA	0x00000000	0x000C311C	87
1.5.2.9	reg : OTP_EFC_WR_DELAY	0x000000A4	0x000C3120	87
1.5.2.10	reg : OTP_EFC_UNUSED	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[0]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[1]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[2]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[3]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[4]	0x00000000	0x000C3124, 0x000C3128...	87

1.5.2.10	reg : OTP_EFC_UNUSED[5]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[6]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[7]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[8]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[9]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[10]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[11]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[12]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[13]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[14]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[15]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[16]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[17]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[18]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[19]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[20]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[21]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[22]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[23]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[24]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[25]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[26]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[27]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[28]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[29]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[30]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[31]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[32]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[33]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[34]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[35]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[36]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[37]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[38]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[39]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[40]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[41]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[42]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[43]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[44]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[45]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[46]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[47]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[48]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[49]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[50]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[51]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[52]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[53]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.2.10	reg : OTP_EFC_UNUSED[54]	0x00000000	0x000C3124, 0x000C3128...	87
1.5.3	reg : OTP_LOCK	0x00000000	0x000C3200	87
1.6	section : ROT_KAM		0x000C4000 - 0x000C4703	88
1.6.1	section : KAM_PMR_LIST		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[0]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[1]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[2]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[3]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[4]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[5]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[6]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1	section : KAM_PMR_LIST[7]		0x000C4000, 0x000C4040 ... 0x000C41FF	88
1.6.1.1	reg : KAM_PMR_ENTRY	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[0]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[1]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[2]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[3]	0x00000000	0x000C4000, 0x000C4004...	88

1.6.1.1	reg : KAM_PMR_ENTRY[4]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[5]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[6]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[7]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[8]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[9]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[10]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[11]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[12]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[13]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[14]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.1.1	reg : KAM_PMR_ENTRY[15]	0x00000000	0x000C4000, 0x000C4004...	88
1.6.2	reg : KAM_PMR_STATUS	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[0]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[1]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[2]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[3]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[4]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[5]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[6]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.2	reg : KAM_PMR_STATUS[7]	0x00000000	0x000C4200, 0x000C4204...	88
1.6.3	reg : KAM_PMR_SOFT_RESET	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[0]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[1]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[2]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[3]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[4]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[5]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[6]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.3	reg : KAM_PMR_SOFT_RESET[7]	0x00000000	0x000C4220, 0x000C4224...	89
1.6.4	reg : KAM_PMR_SOFT_RESET_LOK	0x00000000	0x000C4240	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[0]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[1]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[2]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[3]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[4]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[5]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[6]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.5	reg : KAM_PMR_EXTEND_LOCK[7]	0x00000000	0x000C4244, 0x000C4248...	89
1.6.6	reg : KAM_PMR_MEASUREMENT	0x00000000	0x000C4264	89
1.6.7	reg : KAM_PMR_EXTEND_CTRL	0x00000000	0x000C4268	89
1.6.8	reg : KAM_PMR_EXTEND_STATUS	0x00000201	0x000C426C	90
1.6.9	reg : KAM_PMR_BUS_ERR_STATU	0x00000000	0x000C4270	90

1.6.10	reg : KAM_PMR_SHA_CONFIG	0x00000000	0x000C4274	90
1.6.11	reg : KAM_PMR_SHA_INTR	0x00000000	0x000C4278	91
1.6.12	reg : KAM_ECC_CTRL_PMR	0x00000000	0x000C427C	92
1.6.13	reg : KAM_ECC_STATUS_PMR	0x0000F000	0x000C4280	92
1.6.14	reg : KAM_ECC_LOG_PMR	0x00000000	0x000C4284	93
1.6.15	section : KAM_OTP_LANE		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[0]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[1]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[2]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[3]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[4]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[5]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[6]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15	section : KAM_OTP_LANE[7]		0x000C4300, 0x000C4310 ... 0x000C437B	93
1.6.15.1	reg : CONFIG	0x00000000	0x000C4300, 0x0000000000...	93
1.6.15.2	reg : ACCESS	0x00000003	0x000C4304, 0x0000000000...	94
1.6.15.3	reg : STATUS	0x00000000	0x000C4308, 0x0000000000...	94
1.6.16	reg : KAM_OTP_HW_CONTROL_VA ID	0x00000000	0x000C43C0	94
1.6.17	reg : KAM_OTP_HW_CONTROL	0x00001FFF	0x000C43C4	95
1.6.18	section : KAM_KV		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[0]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[1]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[2]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[3]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[4]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18	section : KAM_KV[5]		0x000C4400, 0x000C440C ... 0x000C4447	96
1.6.18.1	reg : RANGE	0x00000000	0x000C4400, 0x0000000000...	96
1.6.18.2	reg : ACCESS	0x00000003	0x000C4404, 0x0000000000...	97
1.6.18.3	reg : STATUS	0x00000000	0x000C4408, 0x0000000000...	97
1.6.19	reg : KAM_ERR_STATUS	0x00000000	0x000C4500	97
1.6.20	reg : KAM_INTR_STATE	0x00000000	0x000C4504	98
1.6.21	reg : KAM_INTR_ENABLE	0x00000003	0x000C4508	99
1.6.22	reg : KAM_INTR_TEST	0x00000000	0x000C450C	100
1.6.23	reg : KAM_OTP_BUS_ERR_STATU	0x00000000	0x000C4510	100
1.6.24	reg : KAM_SCRATCH	0x00000000	0x000C4700	100
1.7	section : ROT_SHA		0x000D2000 - 0x000D221F	100
1.7.1	reg : SHA_CTRL	0x00000000	0x000D2000	100
1.7.2	reg : SHA_CFG	0x01010000	0x000D2004	101
1.7.3	reg : SHA_MSG	0x00000000	0x000D2008	102
1.7.4	reg : SHA_GEN_DIGEST	0x00000000	0x000D200C	103
1.7.5	reg : SHA_GEN_DIGEST_BE	0x0000000F	0x000D2010	104
1.7.6	reg : SHA_GEN_CONTEXT	0x00000000	0x000D2014	105
1.7.7	reg : SHA_VERIFY	0x00000000	0x000D2018	105
1.7.8	reg : SHA_MSG_LEN	0x00000000	0x000D2020, 0x000D2024...	106
1.7.8	reg : SHA_MSG_LEN[0]	0x00000000	0x000D2020, 0x000D2024...	106
1.7.8	reg : SHA_MSG_LEN[1]	0x00000000	0x000D2020, 0x000D2024...	106
1.7.8	reg : SHA_MSG_LEN[2]	0x00000000	0x000D2020, 0x000D2024...	106
1.7.8	reg : SHA_MSG_LEN[3]	0x00000000	0x000D2020, 0x000D2024...	106
1.7.9	reg : SHA_DIGEST	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[0]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[1]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[2]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[3]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[4]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[5]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[6]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[7]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[8]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[9]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[10]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[11]	0x00000000	0x000D2030, 0x000D2034...	106

1.7.9	reg : SHA_DIGEST[12]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[13]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[14]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.9	reg : SHA_DIGEST[15]	0x00000000	0x000D2030, 0x000D2034...	106
1.7.10	reg : SHA_STATUS	0x00800001	0x000D2070	107
1.7.11	reg : SHA_ERR_STATUS	0x00000000	0x000D2074	107
1.7.12	reg : SHA_ALERT_STATUS	0x00000000	0x000D2078	108
1.7.13	reg : SHA_INTR_STATE	0x00000004	0x000D2080	109
1.7.14	reg : SHA_INTR_ENABLE	0x00000003	0x000D2084	109
1.7.15	reg : SHA_INTR_TEST	0x00000000	0x000D2088	110
1.7.16	reg : SHA_DBG	0x00002000	0x000D2090	111
1.7.17	reg : SHA_KEY	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[0]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[1]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[2]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[3]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[4]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[5]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[6]	0x00000000	0x000D2200, 0x000D2204...	112
1.7.17	reg : SHA_KEY[7]	0x00000000	0x000D2200, 0x000D2204...	112
1.8	section : ROT_AES		0x000D3000 - 0x000D321F	113
1.8.1	reg : AES_CTRL	0x00000000	0x000D3000	113
1.8.2	reg : AES_CFG	0x10100040	0x000D3004	113
1.8.3	reg : AES_P_LEN_0	0x00000000	0x000D3008	114
1.8.4	reg : AES_P_LEN_1	0x00000000	0x000D300C	115
1.8.5	reg : AES_AAD_LEN	0x00000000	0x000D3010, 0x000D3014...	115
1.8.5	reg : AES_AAD_LEN[0]	0x00000000	0x000D3010, 0x000D3014...	115
1.8.5	reg : AES_AAD_LEN[1]	0x00000000	0x000D3010, 0x000D3014...	115
1.8.6	reg : AES_IV	0x00000000	0x000D3018, 0x000D301C...	115
1.8.6	reg : AES_IV[0]	0x00000000	0x000D3018, 0x000D301C...	115
1.8.6	reg : AES_IV[1]	0x00000000	0x000D3018, 0x000D301C...	115
1.8.6	reg : AES_IV[2]	0x00000000	0x000D3018, 0x000D301C...	115
1.8.6	reg : AES_IV[3]	0x00000000	0x000D3018, 0x000D301C...	115
1.8.7	reg : AES_DATA_IN	0x00000000	0x000D3028	115
1.8.8	reg : AES_DATA_OUT	0x00000000	0x000D302C	116
1.8.9	reg : AES_STATUS	0x00200001	0x000D3040	116
1.8.10	reg : AES_ERR_STATUS	0x00000000	0x000D3044	116
1.8.11	reg : AES_ALERT_STATUS	0x00000000	0x000D3048	116
1.8.12	reg : AES_INTR_STATE	0x00000004	0x000D3050	117
1.8.13	reg : AES_INTR_ENABLE	0x00000003	0x000D3054	117
1.8.14	reg : AES_INTR_TEST	0x00000000	0x000D3058	118
1.8.15	reg : AES_DEBUG	0x00000408	0x000D3064	118
1.8.16	reg : AES_DATA_OUT_POP	0x00000000	0x000D311C	119
1.8.17	reg : AES_KEY	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[0]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[1]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[2]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[3]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[4]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[5]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[6]	0x00000000	0x000D3200, 0x000D3204...	119
1.8.17	reg : AES_KEY[7]	0x00000000	0x000D3200, 0x000D3204...	119
1.9	section : GCE_REGS_SYS		0x000D3800 - 0x000D38F3	119
1.9.1	section : gce_regs		0x000D3800 - 0x000D38F3	119
1.9.1.1	reg : GCE_AES	0x00000000	0x000D3800	120
1.9.1.2	reg : GCE_CTRL	0x00000000	0x000D3808	120
1.9.1.3	reg : GCE_CFG	0x00104000	0x000D3810	120
1.9.1.4	reg : GCE_IV0	0x00000000	0x000D3818	120
1.9.1.5	reg : GCE_IV1	0x00000000	0x000D3820	120
1.9.1.6	reg : GCE_IV2	0x00000000	0x000D3828	120
1.9.1.7	reg : GCE_IV3	0x00000000	0x000D3830	120
1.9.1.8	reg : GCE_KEY0	0x00000000	0x000D3838	120
1.9.1.9	reg : GCE_KEY1	0x00000000	0x000D3840	121
1.9.1.10	reg : GCE_KEY2	0x00000000	0x000D3848	121
1.9.1.11	reg : GCE_KEY3	0x00000000	0x000D3850	121
1.9.1.12	reg : GCE_KEY4	0x00000000	0x000D3858	121

1.9.1.13	reg : GCE_KEY5	0x00000000	0x000D3860	121
1.9.1.14	reg : GCE_KEY6	0x00000000	0x000D3868	121
1.9.1.15	reg : GCE_KEY7	0x00000000	0x000D3870	121
1.9.1.16	reg : GCE_DATA_IN	0x00000000	0x000D3878	121
1.9.1.17	reg : GCE_DATA_OUT	0x00000000	0x000D3880	121
1.9.1.18	reg : GCE_STATUS	0x00000081	0x000D3888	121
1.9.1.19	reg : GCE_ERR_STATUS	0x00000000	0x000D3890	122
1.9.1.20	reg : GCE_ALERT_STATUS	0x00000000	0x000D3898	122
1.9.1.21	reg : GCE_INTR_STATE	0x00000004	0x000D38A0	122
1.9.1.22	reg : GCE_INTR_ENABLE	0x00000003	0x000D38A8	122
1.9.1.23	reg : GCE_DATA_OUT_POP	0x00000000	0x000D38B0	122
1.9.1.24	reg : GCE_ADD_LENGTH	0x00000000	0x000D38B8	122
1.9.1.25	reg : GCE_LAST	0x00000000	0x000D38C0	123
1.9.1.26	reg : GCE_BCN	0x00000000	0x000D38C8	123
1.9.1.27	reg : GCE_QBCN	0x00000000	0x000D38D0	123
1.9.1.28	reg : GCE_TAG0	0x00000000	0x000D38D8	123
1.9.1.29	reg : GCE_TAG1	0x00000000	0x000D38E0	123
1.9.1.30	reg : GCE_TAG2	0x00000000	0x000D38E8	123
1.9.1.31	reg : GCE_TAG3	0x00000000	0x000D38F0	123
1.10	section : ROT_ECA		0x000D4000 - 0x000D4E77	123
1.10.1	reg : ECA_MEM	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[0]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[1]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[2]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[3]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[4]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[5]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[6]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[7]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[8]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[9]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[10]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[11]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[12]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[13]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[14]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[15]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[16]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[17]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[18]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[19]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[20]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[21]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[22]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[23]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[24]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[25]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[26]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[27]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[28]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[29]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[30]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[31]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[32]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[33]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[34]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[35]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[36]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[37]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[38]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[39]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[40]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[41]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[42]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[43]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[44]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[45]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[46]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[47]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[48]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[49]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[50]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[51]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[52]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[53]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[54]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[55]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[56]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[57]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[58]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[59]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[60]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[61]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[62]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[63]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[64]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[65]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[66]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[67]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[68]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[69]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[70]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[71]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[72]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[73]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[74]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[75]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[76]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[77]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[78]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[79]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[80]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[81]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[82]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[83]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[84]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[85]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[86]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[87]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[88]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[89]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[90]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[91]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[92]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[93]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[94]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[95]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[96]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[97]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[98]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[99]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[100]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[101]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[102]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[103]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[104]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[105]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[106]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[107]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[108]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[109]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[110]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[111]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[112]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[113]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[114]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[115]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[116]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[117]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[118]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[119]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[120]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[121]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[122]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[123]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[124]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[125]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[126]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[127]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[128]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[129]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[130]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[131]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[132]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[133]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[134]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[135]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[136]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[137]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[138]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[139]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[140]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[141]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[142]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[143]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[144]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[145]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[146]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[147]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[148]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[149]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[150]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[151]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[152]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[153]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[154]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[155]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[156]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[157]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[158]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[159]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[160]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[161]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[162]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[163]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[164]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[165]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[166]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[167]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[168]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[169]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[170]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[171]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[172]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[173]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[174]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[175]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[176]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[177]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[178]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[179]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[180]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[181]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[182]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[183]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[184]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[185]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[186]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[187]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[188]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[189]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[190]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[191]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[192]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[193]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[194]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[195]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[196]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[197]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[198]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[199]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[200]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[201]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[202]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[203]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[204]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[205]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[206]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[207]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[208]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[209]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[210]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[211]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[212]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[213]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[214]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[215]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[216]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[217]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[218]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[219]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[220]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[221]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[222]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[223]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[224]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[225]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[226]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[227]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[228]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[229]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[230]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[231]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[232]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[233]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[234]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[235]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[236]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[237]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[238]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[239]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[240]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[241]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[242]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[243]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[244]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[245]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[246]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[247]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[248]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[249]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[250]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[251]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[252]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[253]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[254]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[255]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[256]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[257]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[258]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[259]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[260]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[261]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[262]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[263]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[264]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[265]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[266]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[267]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[268]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[269]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[270]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[271]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[272]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[273]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[274]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[275]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[276]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[277]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[278]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[279]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[280]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[281]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[282]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[283]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[284]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[285]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[286]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[287]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[288]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[289]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[290]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[291]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[292]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[293]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[294]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[295]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[296]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[297]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[298]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[299]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[300]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[301]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[302]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[303]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[304]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[305]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[306]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[307]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[308]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[309]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[310]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[311]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[312]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[313]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[314]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[315]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[316]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[317]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[318]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[319]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[320]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[321]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[322]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[323]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[324]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[325]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[326]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[327]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[328]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[329]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[330]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[331]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[332]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[333]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[334]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[335]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[336]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[337]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[338]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[339]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[340]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[341]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[342]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[343]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[344]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[345]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[346]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[347]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[348]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[349]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[350]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[351]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[352]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[353]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[354]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[355]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[356]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[357]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[358]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[359]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[360]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[361]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[362]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[363]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[364]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[365]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[366]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[367]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[368]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[369]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[370]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[371]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[372]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[373]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[374]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[375]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[376]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[377]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[378]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[379]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[380]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[381]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[382]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[383]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[384]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[385]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[386]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[387]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[388]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[389]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[390]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[391]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[392]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[393]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[394]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[395]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[396]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[397]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[398]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[399]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[400]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[401]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[402]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[403]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[404]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[405]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[406]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[407]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[408]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[409]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[410]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[411]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[412]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[413]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[414]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[415]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[416]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[417]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[418]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[419]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[420]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[421]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[422]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[423]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[424]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[425]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[426]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[427]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[428]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[429]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[430]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[431]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[432]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[433]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[434]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[435]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[436]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[437]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[438]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[439]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[440]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[441]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[442]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[443]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[444]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[445]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[446]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[447]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[448]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[449]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[450]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[451]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[452]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[453]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[454]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[455]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[456]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[457]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[458]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[459]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[460]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[461]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[462]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[463]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[464]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[465]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[466]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[467]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[468]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[469]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[470]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[471]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[472]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[473]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[474]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[475]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[476]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[477]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[478]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[479]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[480]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[481]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[482]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[483]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[484]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[485]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[486]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[487]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[488]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[489]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[490]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[491]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[492]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[493]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[494]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[495]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[496]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[497]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[498]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[499]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[500]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[501]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[502]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[503]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[504]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[505]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[506]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[507]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[508]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[509]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[510]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[511]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[512]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[513]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[514]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[515]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[516]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[517]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[518]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[519]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[520]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[521]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[522]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[523]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[524]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[525]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[526]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[527]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[528]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[529]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[530]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[531]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[532]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[533]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[534]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[535]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[536]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[537]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[538]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[539]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[540]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[541]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[542]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[543]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[544]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[545]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[546]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[547]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[548]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[549]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[550]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[551]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[552]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[553]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[554]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[555]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[556]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[557]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[558]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[559]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[560]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[561]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[562]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[563]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[564]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[565]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[566]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[567]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[568]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[569]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[570]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[571]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[572]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[573]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[574]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[575]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[576]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[577]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[578]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[579]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[580]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[581]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[582]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[583]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[584]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[585]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[586]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[587]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[588]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[589]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[590]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[591]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[592]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[593]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[594]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[595]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[596]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[597]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[598]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[599]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[600]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[601]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[602]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[603]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[604]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[605]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[606]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[607]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[608]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[609]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[610]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[611]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[612]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[613]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[614]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[615]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[616]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[617]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[618]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[619]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[620]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[621]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[622]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[623]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[624]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[625]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[626]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[627]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[628]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[629]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[630]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[631]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[632]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[633]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[634]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[635]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[636]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[637]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[638]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[639]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[640]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[641]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[642]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[643]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[644]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[645]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[646]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[647]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[648]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[649]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[650]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[651]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[652]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[653]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[654]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[655]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[656]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[657]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[658]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[659]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[660]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[661]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[662]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[663]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[664]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[665]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[666]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[667]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[668]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[669]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[670]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[671]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[672]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[673]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[674]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[675]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[676]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[677]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[678]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[679]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[680]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[681]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[682]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[683]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[684]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[685]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[686]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[687]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[688]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[689]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[690]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[691]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[692]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[693]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[694]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[695]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[696]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[697]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[698]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[699]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[700]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[701]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[702]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[703]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[704]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[705]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[706]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[707]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[708]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[709]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[710]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[711]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[712]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[713]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[714]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[715]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[716]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[717]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[718]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[719]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[720]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[721]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[722]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[723]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[724]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[725]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[726]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[727]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[728]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[729]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[730]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[731]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[732]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[733]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[734]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[735]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[736]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[737]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[738]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[739]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[740]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[741]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[742]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[743]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[744]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[745]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[746]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[747]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[748]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[749]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[750]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[751]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[752]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[753]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[754]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[755]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[756]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[757]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[758]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[759]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[760]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[761]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[762]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[763]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[764]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[765]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[766]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[767]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[768]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[769]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[770]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[771]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[772]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[773]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[774]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[775]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[776]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[777]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[778]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[779]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[780]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[781]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[782]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[783]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[784]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[785]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[786]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[787]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[788]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[789]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[790]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[791]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[792]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[793]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[794]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[795]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[796]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[797]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[798]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[799]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[800]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[801]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[802]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[803]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[804]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[805]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[806]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[807]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[808]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[809]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[810]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[811]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[812]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[813]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[814]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[815]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[816]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[817]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[818]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[819]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[820]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[821]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[822]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[823]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[824]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[825]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[826]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[827]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[828]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[829]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[830]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[831]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[832]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[833]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[834]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[835]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[836]	0x00000000	0x000D4000, 0x000D4004...	124

1.10.1	reg : ECA_MEM[837]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[838]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[839]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[840]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[841]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[842]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[843]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[844]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[845]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[846]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[847]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[848]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[849]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[850]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[851]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[852]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[853]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[854]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[855]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[856]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[857]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[858]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[859]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[860]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[861]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[862]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[863]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[864]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[865]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[866]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[867]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[868]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[869]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[870]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[871]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[872]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[873]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[874]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[875]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[876]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[877]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[878]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[879]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[880]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[881]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[882]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[883]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[884]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[885]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[886]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[887]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[888]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[889]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[890]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[891]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[892]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[893]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[894]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.1	reg : ECA_MEM[895]	0x00000000	0x000D4000, 0x000D4004...	124
1.10.2	reg : ECA_CTRL	0x00000000	0x000D4E00	124
1.10.3	reg : ECA_CFG	0x00000000	0x000D4E04	125
1.10.4	reg : ECA_P_UNCOMP	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[0]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[1]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[2]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[3]	0x00000000	0x000D4E08, 0x000D4E0C...	125

1.10.4	reg : ECA_P_UNCOMP[4]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[5]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[6]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[7]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[8]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[9]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[10]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[11]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[12]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[13]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[14]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.4	reg : ECA_P_UNCOMP[15]	0x00000000	0x000D4E08, 0x000D4E0C...	125
1.10.5	reg : ECA_P_COEF_0	0x00000000	0x000D4E48	125
1.10.6	reg : ECA_P_COEF_1	0x00000000	0x000D4E4C	126
1.10.7	reg : ECA_P_PARAMS	0x00000000	0x000D4E50	126
1.10.8	reg : ECA_STATUS	0x00000001	0x000D4E54	126
1.10.9	reg : ECA_ERR_STATUS	0x00000000	0x000D4E58	126
1.10.10	reg : ECA_ALERT_STATUS	0x00000000	0x000D4E5C	127
1.10.11	reg : ECA_INTR_STATE	0x00000000	0x000D4E60	127
1.10.12	reg : ECA_INTR_ENABLE	0x00000003	0x000D4E64	127
1.10.13	reg : ECA_INTR_TEST	0x00000000	0x000D4E68	128
1.10.14	reg : ECA_MEM_TEST0	0x00000000	0x000D4E6C	128
1.10.15	reg : ECA_MEM_TEST1	0x00000000	0x000D4E70	129
1.10.16	reg : ECA_DEBUG	0x00000000	0x000D4E74	129
1.11	section : ROT_MAA		0x000D5000 - 0x000D5E2B	129
1.11.1	reg : MAA_SEG_1	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[0]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[1]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[2]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[3]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[4]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[5]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[6]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[7]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[8]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[9]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[10]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[11]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[12]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[13]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[14]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[15]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[16]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[17]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[18]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[19]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[20]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[21]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[22]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[23]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[24]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[25]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[26]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[27]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[28]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[29]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[30]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[31]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[32]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[33]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[34]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[35]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[36]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[37]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[38]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[39]	0x00000000	0x000D5000, 0x000D5004...	129

1.11.1	reg : MAA_SEG_1[40]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[41]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[42]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[43]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[44]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[45]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[46]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[47]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[48]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[49]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[50]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[51]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[52]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[53]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[54]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[55]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[56]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[57]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[58]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[59]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[60]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[61]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[62]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[63]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[64]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[65]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[66]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[67]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[68]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[69]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[70]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[71]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[72]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[73]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[74]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[75]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[76]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[77]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[78]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[79]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[80]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[81]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[82]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[83]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[84]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[85]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[86]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[87]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[88]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[89]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[90]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[91]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[92]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[93]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[94]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[95]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[96]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[97]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[98]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[99]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[100]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[101]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[102]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[103]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[104]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[105]	0x00000000	0x000D5000, 0x000D5004...	129

1.11.1	reg : MAA_SEG_1[106]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[107]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[108]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[109]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[110]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[111]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[112]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[113]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[114]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[115]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[116]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[117]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[118]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[119]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[120]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[121]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[122]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[123]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[124]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[125]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[126]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.1	reg : MAA_SEG_1[127]	0x00000000	0x000D5000, 0x000D5004...	129
1.11.2	reg : MAA_SEG_2	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[0]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[1]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[2]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[3]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[4]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[5]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[6]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[7]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[8]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[9]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[10]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[11]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[12]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[13]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[14]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[15]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[16]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[17]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[18]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[19]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[20]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[21]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[22]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[23]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[24]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[25]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[26]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[27]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[28]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[29]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[30]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[31]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[32]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[33]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[34]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[35]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[36]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[37]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[38]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[39]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[40]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[41]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[42]	0x00000000	0x000D5200, 0x000D5204...	130

1.11.2	reg : MAA_SEG_2[43]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[44]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[45]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[46]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[47]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[48]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[49]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[50]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[51]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[52]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[53]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[54]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[55]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[56]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[57]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[58]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[59]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[60]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[61]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[62]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[63]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[64]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[65]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[66]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[67]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[68]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[69]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[70]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[71]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[72]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[73]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[74]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[75]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[76]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[77]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[78]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[79]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[80]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[81]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[82]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[83]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[84]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[85]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[86]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[87]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[88]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[89]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[90]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[91]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[92]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[93]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[94]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[95]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[96]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[97]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[98]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[99]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[100]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[101]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[102]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[103]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[104]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[105]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[106]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[107]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[108]	0x00000000	0x000D5200, 0x000D5204...	130

1.11.2	reg : MAA_SEG_2[109]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[110]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[111]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[112]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[113]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[114]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[115]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[116]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[117]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[118]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[119]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[120]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[121]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[122]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[123]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[124]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[125]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[126]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.2	reg : MAA_SEG_2[127]	0x00000000	0x000D5200, 0x000D5204...	130
1.11.3	reg : MAA_SEG_3	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[0]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[1]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[2]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[3]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[4]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[5]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[6]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[7]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[8]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[9]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[10]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[11]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[12]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[13]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[14]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[15]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[16]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[17]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[18]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[19]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[20]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[21]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[22]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[23]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[24]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[25]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[26]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[27]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[28]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[29]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[30]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[31]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[32]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[33]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[34]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[35]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[36]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[37]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[38]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[39]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[40]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[41]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[42]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[43]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[44]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[45]	0x00000000	0x000D5400, 0x000D5404...	130

1.11.3	reg : MAA_SEG_3[46]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[47]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[48]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[49]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[50]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[51]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[52]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[53]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[54]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[55]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[56]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[57]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[58]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[59]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[60]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[61]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[62]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[63]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[64]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[65]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[66]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[67]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[68]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[69]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[70]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[71]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[72]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[73]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[74]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[75]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[76]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[77]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[78]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[79]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[80]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[81]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[82]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[83]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[84]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[85]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[86]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[87]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[88]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[89]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[90]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[91]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[92]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[93]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[94]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[95]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[96]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[97]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[98]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[99]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[100]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[101]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[102]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[103]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[104]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[105]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[106]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[107]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[108]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[109]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[110]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[111]	0x00000000	0x000D5400, 0x000D5404...	130

1.11.3	reg : MAA_SEG_3[112]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[113]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[114]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[115]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[116]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[117]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[118]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[119]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[120]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[121]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[122]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[123]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[124]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[125]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[126]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.3	reg : MAA_SEG_3[127]	0x00000000	0x000D5400, 0x000D5404...	130
1.11.4	reg : MAA_SEG_4	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[0]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[1]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[2]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[3]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[4]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[5]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[6]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[7]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[8]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[9]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[10]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[11]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[12]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[13]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[14]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[15]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[16]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[17]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[18]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[19]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[20]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[21]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[22]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[23]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[24]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[25]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[26]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[27]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[28]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[29]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[30]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[31]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[32]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[33]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[34]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[35]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[36]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[37]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[38]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[39]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[40]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[41]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[42]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[43]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[44]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[45]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[46]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[47]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[48]	0x00000000	0x000D5600, 0x000D5604...	130

1.11.4	reg : MAA_SEG_4[49]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[50]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[51]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[52]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[53]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[54]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[55]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[56]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[57]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[58]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[59]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[60]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[61]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[62]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[63]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[64]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[65]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[66]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[67]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[68]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[69]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[70]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[71]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[72]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[73]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[74]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[75]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[76]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[77]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[78]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[79]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[80]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[81]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[82]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[83]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[84]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[85]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[86]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[87]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[88]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[89]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[90]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[91]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[92]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[93]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[94]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[95]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[96]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[97]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[98]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[99]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[100]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[101]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[102]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[103]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[104]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[105]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[106]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[107]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[108]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[109]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[110]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[111]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[112]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[113]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[114]	0x00000000	0x000D5600, 0x000D5604...	130

1.11.4	reg : MAA_SEG_4[115]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[116]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[117]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[118]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[119]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[120]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[121]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[122]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[123]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[124]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[125]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[126]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.4	reg : MAA_SEG_4[127]	0x00000000	0x000D5600, 0x000D5604...	130
1.11.5	reg : MAA_SEG_5	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[0]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[1]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[2]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[3]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[4]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[5]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[6]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[7]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[8]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[9]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[10]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[11]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[12]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[13]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[14]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[15]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[16]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[17]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[18]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[19]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[20]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[21]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[22]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[23]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[24]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[25]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[26]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[27]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[28]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[29]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[30]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[31]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[32]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[33]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[34]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[35]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[36]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[37]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[38]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[39]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[40]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[41]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[42]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[43]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[44]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[45]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[46]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[47]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[48]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[49]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[50]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[51]	0x00000000	0x000D5800, 0x000D5804...	130

1.11.5	reg : MAA_SEG_5[52]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[53]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[54]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[55]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[56]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[57]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[58]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[59]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[60]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[61]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[62]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[63]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[64]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[65]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[66]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[67]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[68]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[69]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[70]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[71]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[72]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[73]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[74]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[75]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[76]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[77]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[78]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[79]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[80]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[81]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[82]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[83]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[84]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[85]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[86]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[87]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[88]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[89]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[90]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[91]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[92]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[93]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[94]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[95]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[96]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[97]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[98]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[99]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[100]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[101]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[102]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[103]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[104]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[105]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[106]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[107]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[108]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[109]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[110]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[111]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[112]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[113]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[114]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[115]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[116]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[117]	0x00000000	0x000D5800, 0x000D5804...	130

1.11.5	reg : MAA_SEG_5[118]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[119]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[120]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[121]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[122]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[123]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[124]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[125]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[126]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.5	reg : MAA_SEG_5[127]	0x00000000	0x000D5800, 0x000D5804...	130
1.11.6	reg : MAA_SEG_6	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[0]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[1]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[2]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[3]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[4]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[5]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[6]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[7]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[8]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[9]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[10]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[11]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[12]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[13]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[14]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[15]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[16]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[17]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[18]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[19]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[20]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[21]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[22]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[23]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[24]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[25]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[26]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[27]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[28]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[29]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[30]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[31]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[32]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[33]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[34]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[35]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[36]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[37]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[38]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[39]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[40]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[41]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[42]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[43]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[44]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[45]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[46]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[47]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[48]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[49]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[50]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[51]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[52]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[53]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[54]	0x00000000	0x000D5A00, 0x000D5A04...	131

1.11.6	reg : MAA_SEG_6[55]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[56]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[57]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[58]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[59]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[60]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[61]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[62]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[63]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[64]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[65]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[66]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[67]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[68]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[69]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[70]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[71]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[72]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[73]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[74]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[75]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[76]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[77]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[78]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[79]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[80]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[81]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[82]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[83]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[84]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[85]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[86]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[87]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[88]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[89]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[90]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[91]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[92]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[93]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[94]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[95]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[96]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[97]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[98]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[99]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[100]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[101]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[102]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[103]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[104]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[105]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[106]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[107]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[108]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[109]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[110]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[111]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[112]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[113]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[114]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[115]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[116]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[117]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[118]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[119]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[120]	0x00000000	0x000D5A00, 0x000D5A04...	131

1.11.6	reg : MAA_SEG_6[121]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[122]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[123]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[124]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[125]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[126]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.6	reg : MAA_SEG_6[127]	0x00000000	0x000D5A00, 0x000D5A04...	131
1.11.7	reg : MAA_CTRL	0x00000000	0x000D5E00	131
1.11.8	reg : MAA_CFG	0x00000000	0x000D5E04	132
1.11.9	reg : MAA_STATUS	0x00000001	0x000D5E08	133
1.11.10	reg : MAA_ERR_STATUS	0x00000000	0x000D5E0C	134
1.11.11	reg : MAA_ALERT_STATUS	0x00000000	0x000D5E10	135
1.11.12	reg : MAA_INTR_STATE	0x00000000	0x000D5E14	135
1.11.13	reg : MAA_INTR_ENABLE	0x00000003	0x000D5E18	135
1.11.14	reg : MAA_INTR_TEST	0x00000000	0x000D5E1C	136
1.11.15	reg : MAA_MEM_TEST0	0x00000000	0x000D5E20	136
1.11.16	reg : MAA_MEM_TEST1	0x00000000	0x000D5E24	137
1.11.17	reg : MAA_DEBUG	0x00000080	0x000D5E28	137
1.12	section : RSA_REGS_SYS		0x000D6000 - 0x000D607B	137
1.12.1	section : rsa_regs		0x000D6000 - 0x000D607B	137
1.12.1.1	reg : RSA_DATA_FOR_MEMORY	0x00000000	0x000D6000	138
1.12.1.2	reg : RSA_MEMORY_WRITE_ADDRSS	0x00000000	0x000D6008	138
1.12.1.3	reg : RSA_CONTROL	0x00000000	0x000D6010	138
1.12.1.4	reg : RSA_STATUS	0x00000000	0x000D6018	138
1.12.1.5	reg : RSA_MINV	0x00000000	0x000D6020	138
1.12.1.6	reg : RSA_INTERRUPT_ENABLE	0x00000000	0x000D6028	138
1.12.1.7	reg : RSA_CEN_CONTROL	0x00000001	0x000D6030	138
1.12.1.8	reg : RSA_LENGTH	0x00000000	0x000D6040	139
1.12.1.9	reg : RSA_DATA_LENGTH	0x00000000	0x000D6050	139
1.12.1.10	reg : RSA_MONTGOMERY_DATA	0x00000000	0x000D6058	139
1.12.1.11	reg : RSA_MEMORY_SIZE	0x00000000	0x000D6060	139
1.12.1.12	reg : RSA_MULTIPLICATION	0x00000000	0x000D6068	139
1.12.1.13	reg : RSA_MICROPROGRAM	0x00000000	0x000D6070	139
1.12.1.14	reg : RSA_GROUPS	0x00000000	0x000D6078	139
1.13	section : ROT_TRNG		0x000D7000 - 0x000D7447	139
1.13.1	reg : TRNG_RESERVED	0x00000000	0x000D7000	140
1.13.2	reg : TRNG_SCRATCH	0x00000000	0x000D7004	140
1.13.3	reg : TRNG_CMD	0x00000000	0x000D7010	140
1.13.4	reg : TRNG_STS	0x00000010	0x000D7020	140
1.13.5	reg : TRNG_POOL_STS	0x00000000	0x000D7030	141
1.13.6	reg : TRNG_DOUT	0x00000000	0x000D7040	141
1.13.7	reg : TRNG_ROSC_OUT	0x00000000	0x000D7050	141
1.13.8	reg : TRNG_SAMPLED_ROSC_OUT	0x00000000	0x000D7054	141
1.13.9	reg : TRNG_SAMPLED_OVERFLOWCNT	0x00000000	0x000D7058	141
1.13.10	reg : TRNG_REQ_LEN	0x00000000	0x000D7060	141
1.13.11	reg : TRNG_USER_IN_LEN	0x00000000	0x000D7070	141
1.13.12	reg : TRNG_CTRL	0x00000001	0x000D7080	141
1.13.13	reg : TRNG_ROSC_CTRL	0x00040064	0x000D7084	142
1.13.14	reg : TRNG_ROSC_SELFTEST_CTRL	0xFFFFFFFF	0x000D7088	142
1.13.15	reg : TRNG_MIN_ENTR	0x00000004	0x000D7090	142
1.13.16	reg : TRNG_ENTROPY	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[0]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[1]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[2]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[3]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[4]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[5]	0x00000000	0x000D7100, 0x000D7104...	142

1.13.16	reg : TRNG_ENTROPY[6]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.16	reg : TRNG_ENTROPY[7]	0x00000000	0x000D7100, 0x000D7104...	142
1.13.17	reg : TRNG_NONCE	0x00000000	0x000D7120, 0x000D7124...	142
1.13.17	reg : TRNG_NONCE[0]	0x00000000	0x000D7120, 0x000D7124...	142
1.13.17	reg : TRNG_NONCE[1]	0x00000000	0x000D7120, 0x000D7124...	142
1.13.17	reg : TRNG_NONCE[2]	0x00000000	0x000D7120, 0x000D7124...	142
1.13.17	reg : TRNG_NONCE[3]	0x00000000	0x000D7120, 0x000D7124...	142
1.13.18	reg : TRNG_USER_IN	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[0]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[1]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[2]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[3]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[4]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[5]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[6]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[7]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[8]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[9]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[10]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.18	reg : TRNG_USER_IN[11]	0x00000000	0x000D7200, 0x000D7204...	143
1.13.19	reg : TRNG_ISR	0x00000000	0x000D7300	143
1.13.20	reg : TRNG_IER	0x00000000	0x000D7304	143
1.13.21	reg : TRNG_IFR	0x00000000	0x000D7308	143
1.13.22	reg : TRNG_CHAR_SCRATCH	0x00000000	0x000D7400	143
1.13.23	reg : TRNG_CHAR_CMD	0x00000000	0x000D7404	143
1.13.24	reg : TRNG_CHAR_STS	0x00000010	0x000D7408	144
1.13.25	reg : TRNG_CHAR_POOL_STS	0x00000000	0x000D740C	144
1.13.26	reg : TRNG_CHAR_ROSC_CTRL	0x00000000	0x000D7410	144
1.13.27	reg : TRNG_CHAR_MIN_ENTR	0x00000004	0x000D7414	144
1.13.28	reg : TRNG_CHAR_ROSC_OUT	0xFFFFFFFF	0x000D7418	145
1.13.29	reg : TRNG_CHAR_SAMPLED_ROS_OUT	0x00000000	0x000D741C	145
1.13.30	reg : TRNG_CHAR_SAMPLED_OVEFLOW_CNT	0x00000000	0x000D7420	145
1.13.31	reg : TRNG_CHAR_ROSC_SELFTET_CTRL	0xFFFFFFFF	0x000D7424	145
1.13.32	reg : TRNG_CS_CTRL	0x00000000	0x000D7430	145
1.13.33	reg : TRNG_CS_TRIGSEL	0x00000000	0x000D7434	145
1.13.34	reg : TRNG_CS_TRIGPOL	0x00000000	0x000D7438	146
1.13.35	reg : TRNG_CS_ACQ_ADDR	0x00000000	0x000D743C	146
1.13.36	reg : TRNG_CS_ACQ_DATA	0x00000000	0x000D7440	146
1.13.37	reg : TRNG_CS_CURR_ADDR	0x00000000	0x000D7444	146
1.14	section : DWC_TRNG_CORE		0x000D8000 - 0x000D80F7	146
1.14.1	section : DWC_trng_core_nist_trng_controller		0x000D8000 - 0x000D80F7	146
1.14.1.1	reg : CTRL	0x00000000	0x000D8000	146
1.14.1.2	reg : MODE	0x00000161	0x000D8004	147
1.14.1.3	reg : SMODE	0x0000002A	0x000D8008	147
1.14.1.4	reg : STAT	0x00000450	0x000D800C	147
1.14.1.5	reg : IE	0x00000000	0x000D8010	147
1.14.1.6	reg : ISTAT	0x00000000	0x000D8014	147
1.14.1.7	reg : ALARMS	0x00000000	0x000D8018	148
1.14.1.8	reg : COREKIT_REL	0x0000300B	0x000D801C	148
1.14.1.9	reg : FEATURES	0x00000301	0x000D8020	148
1.14.1.10	reg : RAND0	0x00000000	0x000D8024	148
1.14.1.11	reg : RAND1	0x00000000	0x000D8028	148
1.14.1.12	reg : RAND2	0x00000000	0x000D802C	148
1.14.1.13	reg : RAND3	0x00000000	0x000D8030	148

1.14.1.14	reg : NPA_DATA0	0x00000000	0x000D8034	149
1.14.1.15	reg : NPA_DATA1	0x00000000	0x000D8038	149
1.14.1.16	reg : NPA_DATA2	0x00000000	0x000D803C	149
1.14.1.17	reg : NPA_DATA3	0x00000000	0x000D8040	149
1.14.1.18	reg : NPA_DATA4	0x00000000	0x000D8044	149
1.14.1.19	reg : NPA_DATA5	0x00000000	0x000D8048	149
1.14.1.20	reg : NPA_DATA6	0x00000000	0x000D804C	149
1.14.1.21	reg : NPA_DATA7	0x00000000	0x000D8050	149
1.14.1.22	reg : NPA_DATA8	0x00000000	0x000D8054	149
1.14.1.23	reg : NPA_DATA9	0x00000000	0x000D8058	149
1.14.1.24	reg : NPA_DATA10	0x00000000	0x000D805C	150
1.14.1.25	reg : NPA_DATA11	0x00000000	0x000D8060	150
1.14.1.26	reg : NPA_DATA12	0x00000000	0x000D8064	150
1.14.1.27	reg : NPA_DATA13	0x00000000	0x000D8068	150
1.14.1.28	reg : NPA_DATA14	0x00000000	0x000D806C	150
1.14.1.29	reg : NPA_DATA15	0x00000000	0x000D8070	150
1.14.1.30	reg : SEED0	0x00000000	0x000D8074	150
1.14.1.31	reg : SEED1	0x00000000	0x000D8078	150
1.14.1.32	reg : SEED2	0x00000000	0x000D807C	150
1.14.1.33	reg : SEED3	0x00000000	0x000D8080	150
1.14.1.34	reg : SEED4	0x00000000	0x000D8084	151
1.14.1.35	reg : SEED5	0x00000000	0x000D8088	151
1.14.1.36	reg : SEED6	0x00000000	0x000D808C	151
1.14.1.37	reg : SEED7	0x00000000	0x000D8090	151
1.14.1.38	reg : SEED8	0x00000000	0x000D8094	151
1.14.1.39	reg : SEED9	0x00000000	0x000D8098	151
1.14.1.40	reg : SEED10	0x00000000	0x000D809C	151
1.14.1.41	reg : SEED11	0x00000000	0x000D80A0	151
1.14.1.42	reg : TIME_TO_SEED	0x00000000	0x000D80D0	151
1.14.1.43	reg : BUILD_CFG0	0x000076C6	0x000D80F0	151
1.14.1.44	reg : BUILD_CFG1	0x50890104	0x000D80F4	152
1.15	section : ROT_RNG		0x000D8800 - 0x000D8907	152
1.15.1	reg : RNG_ETS_CTRL	0x00000000	0x000D8800	152
1.15.2	reg : RNG_ETS_CLK_CTRL	0x00000000	0x000D8804	153
1.15.3	reg : RNG_DOWN_SAMPLE_RATIO	0x00000000	0x000D8808	153
1.15.4	reg : RNG_ETS_STATUS_STICKY	0x00000000	0x000D880C	153
1.15.5	reg : RNG_ETS_LANE_DATA_0	0x00000000	0x000D8810	154
1.15.6	reg : RNG_ETS_LANE_DATA_1	0x00000000	0x000D8814	154
1.15.7	reg : RNG_ETS_LANE_DATA_2	0x00000000	0x000D8818	154
1.15.8	reg : RNG_ETS_LANE_DATA_3	0x00000000	0x000D881C	154
1.15.9	reg : RNG_ETS_LANE_DATA_4	0x00000000	0x000D8820	154
1.15.10	reg : RNG_ETS_LANE_DATA_5	0x00000000	0x000D8824	154
1.15.11	reg : RNG_ETS_LANE_DATA_6	0x00000000	0x000D8828	155
1.15.12	reg : RNG_ETS_LANE_DATA_7	0x00000000	0x000D882C	155
1.15.13	reg : RNG_ETS_LANE_DATA_8	0x00000000	0x000D8830	155
1.15.14	reg : RNG_ETS_LANE_DATA_9	0x00000000	0x000D8834	155
1.15.15	reg : RNG_ETS_LANE_DATA_10	0x00000000	0x000D8838	155
1.15.16	reg : RNG_ETS_LANE_DATA_11	0x00000000	0x000D883C	155
1.15.17	reg : RNG_ETS_PH_XOR_DATA	0x00000000	0x000D8840	156
1.15.18	reg : RNG_ETS_TH_XOR_DATA	0x00000000	0x000D8844	156

1.15.19	reg : RNG_ETS_LANE_DATA_14	0x00000000	0x000D8848	156
1.15.20	reg : RNG_ETS_LANE_DATA_15	0x00000000	0x000D884C	156
1.15.21	reg : RNG_ETS_LANE_DATA_16	0x00000000	0x000D8850	156
1.15.22	reg : RNG_ETS_LANE_DATA_17	0x00000000	0x000D8854	156
1.15.23	reg : RNG_ETS_LANE_DATA_18	0x00000000	0x000D8858	157
1.15.24	reg : RNG_ETS_LANE_DATA_19	0x00000000	0x000D885C	157
1.15.25	reg : RNG_ETS_LANE_DATA_20	0x00000000	0x000D8860	157
1.15.26	reg : RNG_ETS_LANE_DATA_21	0x00000000	0x000D8864	157
1.15.27	reg : RNG_ETS_LANE_DATA_22	0x00000000	0x000D8868	157
1.15.28	reg : RNG_ETS_LANE_DATA_23	0x00000000	0x000D886C	157
1.15.29	reg : RNG_ETS_LANE_DATA_24	0x00000000	0x000D8870	158
1.15.30	reg : RNG_ETS_LANE_DATA_25	0x00000000	0x000D8874	158
1.15.31	reg : RNG_ETS_LANE_DATA_26	0x00000000	0x000D8878	158
1.15.32	reg : RNG_ETS_LANE_DATA_27	0x00000000	0x000D887C	158
1.15.33	reg : RNG_ETS_LANE_DATA_28	0x00000000	0x000D8880	158
1.15.34	reg : RNG_ETS_POWER_VALID	0x00000000	0x000D8884	158
1.15.35	reg : RNG_RCT_PH_THRSHOLD_0	0x00000000	0x000D8888	158
1.15.36	reg : RNG_RCT_PH_THRSHOLD_1	0x00000000	0x000D888C	159
1.15.37	reg : RNG_RCT_PH_THRSHOLD_2	0x00000000	0x000D8890	159
1.15.38	reg : RNG_RCT_PH_THRSHOLD_3	0x00000000	0x000D8894	159
1.15.39	reg : RNG_RCT_PH_THRSHOLD_4	0x00000000	0x000D8898	159
1.15.40	reg : RNG_RCT_PH_THRSHOLD_5	0x00000000	0x000D889C	159
1.15.41	reg : RNG_RCT_PH_THRSHOLD_6	0x00000000	0x000D88A0	160
1.15.42	reg : RNG_RCT_PH_THRSHOLD_7	0x00000000	0x000D88A4	160
1.15.43	reg : RNG_RCT_PH_THRSHOLD_8	0x00000000	0x000D88A8	160
1.15.44	reg : RNG_APT_PH_THRSHOLD_0	0x00000000	0x000D88AC	160
1.15.45	reg : RNG_APT_PH_THRSHOLD_1	0x00000000	0x000D88B0	160
1.15.46	reg : RNG_APT_PH_THRSHOLD_2	0x00000000	0x000D88B4	160
1.15.47	reg : RNG_APT_PH_THRSHOLD_3	0x00000000	0x000D88B8	161
1.15.48	reg : RNG_APT_PH_THRSHOLD_4	0x00000000	0x000D88BC	161
1.15.49	reg : RNG_APT_PH_THRSHOLD_5	0x00000000	0x000D88C0	161
1.15.50	reg : RNG_APT_PH_THRSHOLD_6	0x00000000	0x000D88C4	161
1.15.51	reg : RNG_APT_PH_THRSHOLD_7	0x00000000	0x000D88C8	161
1.15.52	reg : RNG_APT_PH_THRSHOLD_8	0x00000000	0x000D88CC	161

1.15.53	reg : RNG_RCT_TH_THRSHOLD_0	0x00000000	0x000D88D0	162
1.15.54	reg : RNG_RCT_TH_THRSHOLD_1	0x00000000	0x000D88D4	162
1.15.55	reg : RNG_RCT_TH_THRSHOLD_2	0x00000000	0x000D88D8	162
1.15.56	reg : RNG_APT_TH_THRSHOLD_0	0x00000000	0x000D88DC	162
1.15.57	reg : RNG_APT_TH_THRSHOLD_1	0x00000000	0x000D88E0	162
1.15.58	reg : RNG_APT_TH_THRSHOLD_2	0x00000000	0x000D88E4	162
1.15.59	reg : RNG_RCT_XOR_THRSHOLD	0x00000000	0x000D88E8	163
1.15.60	reg : RNG_APT_XOR_THRSHOLD	0x00000000	0x000D88EC	163
1.15.61	reg : RNG_ERR_STATUS	0x00000000	0x000D88F0	163
1.15.62	reg : RNG_FATAL_STATUS	0x00000000	0x000D88F4	163
1.15.63	reg : RNG_INTR_STATE	0x00000000	0x000D88F8	164
1.15.64	reg : RNG_INTR_ENABLE	0x00000003	0x000D88FC	164
1.15.65	reg : RNG_INTR_TEST	0x00000000	0x000D8900	164
1.15.66	reg : RNG_STATUS	0x00000001	0x000D8904	164
1.16	section : ROT_DMA		0x000E0000 - 0x000E5067	165
1.16.1	section : DMA		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[0]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[1]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[2]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[3]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[4]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1	section : DMA[5]		0x000E0000, 0x000E1000 ... 0x000E5067	165
1.16.1.1	reg : PARAMS	0x00000020	0x000E0000, 0x0000000000...	165
1.16.1.2	reg : CFG	0x00002010	0x000E0004, 0x0000000000...	165
1.16.1.3	reg : CTRL	0x00000010	0x000E0008, 0x0000000000...	166
1.16.1.4	reg : SRC	0x00000000	0x000E000C, 0x0000000000...	167
1.16.1.5	reg : DST	0x00000000	0x000E0010, 0x0000000000...	167
1.16.1.6	reg : LEN	0x00000000	0x000E0014, 0x0000000000...	167
1.16.1.7	reg : MODE	0x00000100	0x000E0018, 0x0000000000...	167
1.16.1.8	reg : FW_FLOW_HANDSHAKE	0x00000000	0x000E001C, 0x0000000000...	171
1.16.1.9	reg : COMPLETION_FIFO	0x00000000	0x000E0020, 0x0000000000...	171
1.16.1.10	reg : FILL_VALUE	0x00000000	0x000E0024, 0x0000000000...	171
1.16.1.11	reg : FILL_FAIL_ADDR	0x00000000	0x000E0028, 0x0000000000...	172
1.16.1.12	reg : FILL_FAIL_COUNT	0x00000000	0x000E002C, 0x0000000000...	172
1.16.1.13	reg : STATUS	0x10000102	0x000E0030, 0x0000000000...	172
1.16.1.14	reg : ERR_STATUS	0x00000000	0x000E0034, 0x0000000000...	173
1.16.1.15	reg : BUS_ERROR_READ_STATUS	0x00000000	0x000E0038, 0x0000000000...	174
1.16.1.16	reg : BUS_ERROR_WRITE_STATUS	0x00000000	0x000E003C, 0x0000000000...	174
1.16.1.17	reg : INTR_STATE	0x00000004	0x000E0040, 0x0000000000...	174
1.16.1.18	reg : INTR_ENABLE	0x00000003	0x000E0044, 0x0000000000...	175
1.16.1.19	reg : INTR_TEST	0x00000000	0x000E0048, 0x0000000000...	175
1.16.1.20	reg : CMD_FIFO_TOP_SRC	0x00000000	0x000E004C, 0x0000000000...	176
1.16.1.21	reg : CMD_FIFO_TOP_DST	0x00000000	0x000E0050, 0x0000000000...	176
1.16.1.22	reg : CMD_FIFO_TOP_LEN	0x00000000	0x000E0054, 0x0000000000...	176
1.16.1.23	reg : CMD_FIFO_TOP_MODE	0x00000000	0x000E0058, 0x0000000000...	176
1.16.1.24	reg : DEBUG	0x00000000	0x000E005C, 0x0000000000...	179
1.16.1.25	reg : SCRATCH	0x00000000	0x000E0060, 0x0000000000...	179
1.16.1.26	reg : BUS_CTRL	0x00773311	0x000E0064, 0x0000000000...	179
1.17	section : DW_APB_I2C		0x000F0000 - 0x000F00FF	179
1.17.1	section : DW_apb_i2c_mem_map _DW_apb_i2c_addr_block1		0x000F0000 - 0x000F00FF	180
1.17.1.1	reg : IC_CON	0x0000007F	0x000F0000	180
1.17.1.2	reg : IC_TAR	0x00001055	0x000F0004	180
1.17.1.3	reg : IC_SAR	0x00000055	0x000F0008	181
1.17.1.4	reg : IC_HS_MADDR	0x00000001	0x000F000C	181

1.17.1.5	reg : IC_DATA_CMD	0x00000000	0x000F0010	181
1.17.1.6	reg : IC_SS_SCL_HCNT	0x00000190	0x000F0014	181
1.17.1.7	reg : IC_SS_SCL_LCNT	0x000001D6	0x000F0018	181
1.17.1.8	reg : IC_FS_SCL_HCNT	0x0000003C	0x000F001C	181
1.17.1.9	reg : IC_FS_SCL_LCNT	0x00000082	0x000F0020	181
1.17.1.10	reg : IC_HS_SCL_HCNT	0x00000006	0x000F0024	182
1.17.1.11	reg : IC_HS_SCL_LCNT	0x00000010	0x000F0028	182
1.17.1.12	reg : IC_INTR_STAT	0x00000000	0x000F002C	182
1.17.1.13	reg : IC_INTR_MASK	0x000048FF	0x000F0030	182
1.17.1.14	reg : IC_RAW_INTR_STAT	0x00000000	0x000F0034	183
1.17.1.15	reg : IC_RX_TL	0x00000000	0x000F0038	183
1.17.1.16	reg : IC_TX_TL	0x00000000	0x000F003C	183
1.17.1.17	reg : IC_CLR_INTR	0x00000000	0x000F0040	184
1.17.1.18	reg : IC_CLR_RX_UNDER	0x00000000	0x000F0044	184
1.17.1.19	reg : IC_CLR_RX_OVER	0x00000000	0x000F0048	184
1.17.1.20	reg : IC_CLR_TX_OVER	0x00000000	0x000F004C	184
1.17.1.21	reg : IC_CLR_RD_REQ	0x00000000	0x000F0050	184
1.17.1.22	reg : IC_CLR_TX_ABRT	0x00000000	0x000F0054	184
1.17.1.23	reg : IC_CLR_RX_DONE	0x00000000	0x000F0058	184
1.17.1.24	reg : IC_CLR_ACTIVITY	0x00000000	0x000F005C	185
1.17.1.25	reg : IC_CLR_STOP_DET	0x00000000	0x000F0060	185
1.17.1.26	reg : IC_CLR_START_DET	0x00000000	0x000F0064	185
1.17.1.27	reg : IC_CLR_GEN_CALL	0x00000000	0x000F0068	185
1.17.1.28	reg : IC_ENABLE	0x00000004	0x000F006C	185
1.17.1.29	reg : IC_STATUS	0x00000006	0x000F0070	185
1.17.1.30	reg : IC_TXFLR	0x00000000	0x000F0074	186
1.17.1.31	reg : IC_RXFLR	0x00000000	0x000F0078	186
1.17.1.32	reg : IC_SDA_HOLD	0x00000001	0x000F007C	186
1.17.1.33	reg : IC_TX_ABRT_SOURCE	0x00000000	0x000F0080	187
1.17.1.34	reg : IC_SLV_DATA_NACK_ONLY	0x00000000	0x000F0084	187
1.17.1.35	reg : IC_DMA_CR	0x00000000	0x000F0088	187
1.17.1.36	reg : IC_DMA_TDLR	0x00000000	0x000F008C	188
1.17.1.37	reg : IC_DMA_RDLR	0x00000000	0x000F0090	188
1.17.1.38	reg : IC_SDA_SETUP	0x00000064	0x000F0094	188
1.17.1.39	reg : IC_ACK_GENERAL_CALL	0x00000001	0x000F0098	188
1.17.1.40	reg : IC_ENABLE_STATUS	0x00000000	0x000F009C	188
1.17.1.41	reg : IC_FS_SPKLEN	0x00000005	0x000F00A0	188
1.17.1.42	reg : IC_HS_SPKLEN	0x00000001	0x000F00A4	188
1.17.1.43	reg : IC_CLR_RESTART_DET	0x00000000	0x000F00A8	188
1.17.1.44	reg : IC_SCL_STUCK_AT_LOW_TIMEOUT	0xFFFFFFFF	0x000F00AC	189
1.17.1.45	reg : IC_SDA_STUCK_AT_LOW_TIMEOUT	0xFFFFFFFF	0x000F00B0	189
1.17.1.46	reg : IC_CLR_SCL_STUCK_DET	0x00000000	0x000F00B4	189
1.17.1.47	reg : IC_DEVICE_ID	0x00000000	0x000F00B8	189
1.17.1.48	reg : REG_TIMEOUT_RST	0x00000008	0x000F00F0	189
1.17.1.49	reg : IC_COMP_PARAM_1	0x000F0FEE	0x000F00F4	189
1.17.1.50	reg : IC_COMP_VERSION	0x3230332A	0x000F00F8	189
1.17.1.51	reg : IC_COMP_TYPE	0x44570140	0x000F00FC	190
1.18	section : DW_APB_UART		0x000F1000 - 0x000F10FF	190
1.18.1	section : uart_memory_map_uart_address_block		0x000F1000 - 0x000F10FF	190
1.18.1.1	reg : RBR	0x00000000	0x000F1000	190
1.18.1.2	reg : IER	0x00000000	0x000F1004	190
1.18.1.3	reg : IIR	0x00000001	0x000F1008	190
1.18.1.4	reg : LCR	0x00000000	0x000F100C	190
1.18.1.5	reg : MCR	0x00000000	0x000F1010	191
1.18.1.6	reg : LSR	0x00000060	0x000F1014	191
1.18.1.7	reg : MSR	0x00000000	0x000F1018	191
1.18.1.8	reg : SCR	0x00000000	0x000F101C	191
1.18.1.9	reg : FAR	0x00000000	0x000F1070	191

1.18.1.10	reg : TFR	0x00000000	0x000F1074	192
1.18.1.11	reg : RFW	0x00000000	0x000F1078	192
1.18.1.12	reg : USR	0x00000000	0x000F107C	192
1.18.1.13	reg : HTX	0x00000000	0x000F10A4	192
1.18.1.14	reg : DMASA	0x00000000	0x000F10A8	192
1.18.1.15	reg : RAR	0x00000000	0x000F10C4	192
1.18.1.16	reg : TAR	0x00000000	0x000F10C8	192
1.18.1.17	reg : LCR_EXT	0x00000000	0x000F10CC	192
1.18.1.18	reg : UART_PROT_LEVEL	0x00000002	0x000F10D0	193
1.18.1.19	reg : REG_TIMEOUT_RST	0x00000008	0x000F10D4	193
1.18.1.20	reg : CPR	0x00013332	0x000F10F4	193
1.18.1.21	reg : UCV	0x3430332A	0x000F10F8	193
1.18.1.22	reg : CTR	0x44570110	0x000F10FC	193
1.19	section : SPI_REGS_SYS		0x000F2000 - 0x000F20C7	194
1.19.1	section : spi_regs		0x000F2000 - 0x000F20C7	194
1.19.1.1	reg : SPI_CONTROL_REG	0x00002007	0x000F2080	194
1.19.1.2	reg : SPI_MASTER_CONTROL_REG	0x00000000	0x000F2084	194
1.19.1.3	reg : SPI_COMMAND_REG	0x00000000	0x000F2088	194
1.19.1.4	reg : SPI_INTERRUPT_STATUS_REG	0x00000000	0x000F208C	194
1.19.1.5	reg : SPI_INTERRUPT_MASK_REG	0x00000000	0x000F2090	195
1.19.1.6	reg : SPI_BYTE_COUNT_REG	0x00000000	0x000F2094	195
1.19.1.7	reg : SPI_DRQ_COUNT_REG	0x00000000	0x000F2098	195
1.19.1.8	reg : SPI_DEBUG_REG	0x00000000	0x000F209C	195
1.19.1.9	reg : DATA_PORT_BASE_REG	0x00000000	0x000F20A0	195
1.19.1.10	reg : DEBUG_PORT_BASE_REG	0x00000000	0x000F20A4	195
1.19.1.11	reg : DEBUG_PORT_WR_PTR_REG	0x00000000	0x000F20A8	195
1.19.1.12	reg : DEBUG_PORT_RD_PTR_REG	0x00000000	0x000F20AC	195
1.19.1.13	reg : DATA_PORT_DATA_REG	0x00000000	0x000F20B0	196
1.19.1.14	reg : DEBUG_PORT_DATA_REG	0x00000000	0x000F20B4	196
1.19.1.15	reg : IRQ_DMARQ_DATA_THRESHOLD_REG	0x00000000	0x000F20B8	196
1.19.1.16	reg : DEBUG_PORT_SIZE_REG	0x00000000	0x000F20BC	196
1.19.1.17	reg : TX_THRESHOLD_REG	0x00000000	0x000F20C0	196
1.19.1.18	reg : DMARQ_DEBUG_THRESHOLD_REG	0x00000000	0x000F20C4	196
1.20	section : GPIO_SYS		0x000F3000 - 0x000F30AB	196
1.20.1	section : gpio		0x000F3000 - 0x000F30AB	196
1.20.1.1	reg : GPIO_PER	0x0003C3FF	0x000F3000	197
1.20.1.2	reg : GPIO_PECR	0x00000000	0x000F3004	197
1.20.1.3	reg : GPIO_PESR	0x00000000	0x000F3008	197
1.20.1.4	reg : GPIO_OER	0x00000000	0x000F3010	197
1.20.1.5	reg : GPIO_OECR	0x00000000	0x000F3014	197
1.20.1.6	reg : GPIO_OESR	0x00000000	0x000F3018	197
1.20.1.7	reg : GPIO_ODR	0x00000000	0x000F3020	197
1.20.1.8	reg : GPIO_ODCR	0x00000000	0x000F3024	197
1.20.1.9	reg : GPIO_ODSR	0x00000000	0x000F3028	197
1.20.1.10	reg : GPIO_PSR	0x0003FFFF	0x000F3030	198
1.20.1.11	reg : GPIO_SPR	0x00000000	0x000F3040	198
1.20.1.12	reg : GPIO_SPCR	0x00000000	0x000F3044	198
1.20.1.13	reg : GPIO_SPSR	0x00000000	0x000F3048	198
1.20.1.14	reg : GPIO_IMR	0x00000000	0x000F3050	198
1.20.1.15	reg : GPIO_IMCR	0x00000000	0x000F3054	198
1.20.1.16	reg : GPIO_IMSR	0x00000000	0x000F3058	198

1.20.1.17	reg : GPIO_NIR	0x00000000	0x000F3060	198
1.20.1.18	reg : GPIO_NICR	0x00000000	0x000F3064	198
1.20.1.19	reg : GPIO_NISR	0x00000000	0x000F3068	199
1.20.1.20	reg : GPIO_PE	0x00000000	0x000F3070	199
1.20.1.21	reg : GPIO_PEC	0x00000000	0x000F3074	199
1.20.1.22	reg : GPIO_PES	0x00000000	0x000F3078	199
1.20.1.23	reg : GPIO_PS	0x00000000	0x000F3080	199
1.20.1.24	reg : GPIO_PSC	0x00000000	0x000F3084	199
1.20.1.25	reg : GPIO_PSS	0x00000000	0x000F3088	199
1.20.1.26	reg : GPIO_STE	0x00000000	0x000F3090	199
1.20.1.27	reg : GPIO_STEC	0x00000000	0x000F3094	200
1.20.1.28	reg : GPIO_STES	0x00000000	0x000F3098	200
1.20.1.29	reg : GPIO_IER	0x0003FFFF	0x000F30A0	200
1.20.1.30	reg : GPIO_IECR	0x00000000	0x000F30A4	200
1.20.1.31	reg : GPIO_IERS	0x00000000	0x000F30A8	200
1.21	section : GPIO_SFR_SYS		0x000F4000 - 0x000F418B	200
1.21.1	section : gpio_sfr		0x000F4000 - 0x000F418B	200
1.21.1.1	reg : GPIO_OE_PA	0x00001C00	0x000F4000	200
1.21.1.2	reg : GPIO_OE_PB	0x00000000	0x000F4008	201
1.21.1.3	reg : GPIO_OUT_PA	0x00000000	0x000F4010	201
1.21.1.4	reg : GPIO_OUT_PB	0x00000000	0x000F4018	201
1.21.1.5	reg : GPIO_IN_PA	0x0003E3C9	0x000F4020	201
1.21.1.6	reg : GPIO_IN_PB	0x0003FFF0	0x000F4028	201
1.21.1.7	reg : GPIO_IE_PA	0x00011C00	0x000F4030	201
1.21.1.8	reg : GPIO_IE_PB	0x00000000	0x000F4038	201
1.21.1.9	reg : GPIO_PE_PA	0x00000000	0x000F4040	201
1.21.1.10	reg : GPIO_PE_PB	0x00000000	0x000F4048	202
1.21.1.11	reg : GPIO_PS_PA	0x00000000	0x000F4050	202
1.21.1.12	reg : GPIO_PS_PB	0x00000000	0x000F4058	202
1.21.1.13	reg : ROT_TRACE_EN	0x00000000	0x000F4100	202
1.21.1.14	reg : HTU_SBS_FRAME_SEL	0x00000000	0x000F4108	202
1.21.1.15	reg : SBS_SVCI2AHB_CONFIG	0x00000000	0x000F4110	202
1.21.1.16	reg : SBS_SVCI2AHB_BASE	0x00000000	0x000F4118	202
1.21.1.17	reg : SBS_SVCI2AHB_OFFSET_0	0x00000000	0x000F4120	202
1.21.1.18	reg : SBS_SVCI2AHB_OFFSET_1	0x00000000	0x000F4128	203
1.21.1.19	reg : SBS_SVCI2AHB_DATA_SEL	0x00000000	0x000F4130	203
1.21.1.20	reg : SBS_AXI2SVCI_CONFIG	0x00000000	0x000F4138	203
1.21.1.21	reg : SBS_AXI2SVCI_BASE	0x00000000	0x000F4140	203
1.21.1.22	reg : SBS_AXI2SVCI_OFFSET_0	0x00000000	0x000F4148	203
1.21.1.23	reg : SBS_AXI2SVCI_OFFSET_1	0x00000000	0x000F4150	203
1.21.1.24	reg : SBS_AXI2SVCI_DATA_SEL	0x00000000	0x000F4158	203
1.21.1.25	reg : HTU_TEST_MUX_MASK_31_0	0x00000000	0x000F4160	203
1.21.1.26	reg : HTU_TEST_MUX_MASK_47_32	0x00000000	0x000F4168	203
1.21.1.27	reg : HTU_SBS_SVCI2AHB_COUNT	0x00000000	0x000F4170	204
1.21.1.28	reg : HTU_SBS_AXI2SVCI_COUNT	0x00000000	0x000F4178	204
1.21.1.29	reg : HTU_TEST_MUX_COUNT	0x00000000	0x000F4180	204
1.21.1.30	reg : SPI_DMA_REQ	0x00000004	0x000F4188	204
1.22	memory : ROT_KV_RAM		0x00101000, 0x00101004 ... 0x00101FFF	204
1.22.1	reg : ROT_KV_RAM	0x00000000	0x00101000, 0x0000000000...	
1.23	memory : ROT_OVERLAY		0x08000000, 0x08000004 ... 0x0FFFFFFF	204
1.23.1	reg : ROT_OVERLAY	0x00000000	0x08000000, 0x0000000000...	

1 SECP

Block 0x00000000 -
0x0FFFFFFF

signals:

Name	Type	Description
SW_Reset	sync	level : high width : 1 type : sync
rot_por_rstn	async	level : low width : 1 type : async
kam_por_rstn	async	level : low width : 1 type : async
sys_por_rstn	async	level : low width : 1 type : async

cheader.enum.name_format : %s%r%f
soc : ROT_1.0

1.1 ROT_ROM

ROT_ROM ▤ 0x00000000,
0x00000004 ...
0x0000FFFF

offset		depth	16384	width	32	default	0x0

1.2 ROT_SECP_RAM1

ROT_SECP_RAM1 ▤ 0x00020000,
0x00020004 ...
0x0004FFFF

offset		depth	49152	width	32	default	0x0

The memory size shown is the maximum memory option provided. This memory includes an optional 32KB of always-on RAM at the start of the memory region (implemented only by CS and CSSD SOCs). CSSD, HDD and ESSD SOCs implement only 128KB of this memory. Access beyond 128KB when the memory does not exist will be ignored. (i.e. writes ignored, reads return 0)decode_size : 0x00030000chapter : 1.14, Security Subsystem, ROT Processor RAMmemwidth : 32'ROT_SECP_RAM1' is an external.'type' : this will create a 'mem'.blockgroup : SECUREPROCESSORrevision : revision: 9aff081'Count' : 'ROT_SECP_RAM1' will repeat '49152' times.

1.3 ROT_SECP_RAM2

ROT_SECP_RAM2 ▤ 0x00050000,
0x00050004 ...
0x0007FFFF

offset		depth	49152	width	32	default	0x0

This memory is only instanced on CS SOCs. Accesses to this memory region from other SOCs are ignored (i.e. writes ignored, reads return 0).decode_size : 0x00030000chapter : 1.15, Security Subsystem, ROT Processor RAMmemwidth : 32'ROT_SECP_RAM2' is an external.'type' : this will create a 'mem'.blockgroup : SECUREPROCESSORrevision : revision: 9aff081'Count' : 'ROT_SECP_RAM2' will repeat '49152' times.

1.4 ROT_SYS

RegGrp 0x000B0000 -
0x000B0707

decode_size : 0x00004000
chapter : 1.2, Security Subsystem, ROT Processor IPC
blockgroup : SECUREPROCESSOR
revision : revision: 7aa1e5b

1.4.1 IPC_ETS_0_ENTRY

Reg. 0x000B0000 -
0x000B003F

rtl.hw_rp : true
display_name : ROT EXTP to SECP IPC Register
'Count' : 'IPC_ETS_0_ENTRY' will repeat '16' times.

count	1	2	3	...	14	15	16

address	0xB0000	0xB0004	0xB0008	...	0xB0034	0xB0038	0xB003C

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	EXTP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, EXTP writes are ignored SECP has read-only access.

## 1.4.2 IPC_ETS_0_CONSUMER	RegGrp	0x000B0040 - 0x000B005B

### 1.4.2.1 CONFIG	Reg.	0x000B0040

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x40	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

### 1.4.2.2 STATUS	Reg.	0x000B0044

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled This status is enabled in INTR_ENABLE register, ORd with other enabled

					error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x0	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x0	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.2.3 CLAIM

Reg. 0x000B0048

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.2.4 CONTROL

Reg. 0x000B004C

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.2.5 INTR_STATE

Reg. 0x000B0050

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED • STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ

					• STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.2.6 INTR_ENABLE

0x000B0054

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL

1.4.2.7 INTR_TEST

0x000B0058

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.3 IPC_STE_0_ENTRY

0x000B0080 - 0x000B009F

rtl.hw_rp : true
display_name : ROT SECP to EXTP IPC Register
'Count' : 'IPC_STE_0_ENTRY' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xB0080	0xB0084	0xB0088	...	0xB0094	0xB0098	0xB009C

bits	name	s/w	h/w	default	description
31:0	DATA	rw	rw	0x0	SECP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, SECP writes are ignored EXTP has read-only access.

1.4.4 IPC_STE_0_PRODUCER

RegGrp 0x000B00A0 - 0x000B00BB

1.4.4.1 CONFIG

Reg. 0x000B00A0

no_reg_tests : true

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x20	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

1.4.4.2 STATUS

Reg. 0x000B00A4

no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled This status is enabled in INTR_ENABLE register, ORd with other enabled

					error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x1	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x20	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.4.3 CLAIM

Reg. 0x000B00A8

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.4.4 CONTROL

Reg. 0x000B00AC

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.4.5 INTR_STATE

Reg. 0x000B00B0

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED • STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ

					• STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.4.6 INTR_ENABLE

Reg. 0x000B00B4

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL dontcompare : true

1.4.4.7 INTR_TEST

Reg. 0x000B00B8

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.5 IPC_ETS_1_ENTRY

Reg. 0x000B00C0 - 0x000B00DF

rtl.hw_rp : true
display_name : ROT EXTP to SECP IPC Register
'Count' : 'IPC_ETS_1_ENTRY' will repeat '8' times.

count	1	2	3	...	6	7	8

address	0xB00C0	0xB00C4	0xB00C8	...	0xB00D4	0xB00D8	0xB00DC

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	EXTP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, EXTP writes are ignored SECP has read-only access.

1.4.6 IPC_ETS_1_CONSUMER

RegGrp 0x000B00E0 - 0x000B00FB

1.4.6.1 CONFIG

Reg. 0x000B00E0

no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x20	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

1.4.6.2 STATUS

Reg. 0x000B00E4

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled

					This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x0	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x0	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.6.3 CLAIM

Reg. 0x000B00E8

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.6.4 CONTROL

Reg. 0x000B00EC

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.6.5 INTR_STATE

Reg. 0x000B00F0

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED

					• STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ • STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.6.6 INTR_ENABLE

Reg. 0x000B00F4

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL

1.4.6.7 INTR_TEST

Reg. 0x000B00F8

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.7 IPC_STE_1_ENTRY

Reg. 0x000B0100 - 0x000B011F

rtl.hw_rp : true
display_name : ROT SECP to EXTP IPC Register
'Count' : 'IPC_STE_1_ENTRY' will repeat '8' times.

count	1	2	3	...	6	7	8

address	0xB0100	0xB0104	0xB0108	...	0xB0114	0xB0118	0xB011C

bits	name	s/w	h/w	default	description
31:0	DATA	rw	rw	0x0	SECP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, SECP writes are ignored EXTP has read-only access.

1.4.8 IPC_STE_1_PRODUCER

RegGrp 0x000B0120 - 0x000B013B

1.4.8.1 CONFIG

Reg. 0x000B0120

no_reg_tests : true

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x20	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

1.4.8.2 STATUS

Reg. 0x000B0124

no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled

					This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x1	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x20	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.8.3 CLAIM

Reg. 0x000B0128

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.8.4 CONTROL

Reg. 0x000B012C

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.8.5 INTR_STATE

Reg. 0x000B0130

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED

					• STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ • STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.8.6 INTR_ENABLE

Reg. 0x000B0134

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL dontcompare : true

1.4.8.7 INTR_TEST

Reg. 0x000B0138

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.9 IPC_ETS_2_ENTRY

Reg. 0x000B0140 - 0x000B015F

rtl.hw_rp : true
display_name : ROT EXTP to SECP IPC Register
'Count' : 'IPC_ETS_2_ENTRY' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xB0140	0xB0144	0xB0148	...	0xB0154	0xB0158	0xB015C

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	EXTP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, EXTP writes are ignored SECP has read-only access.

1.4.10 IPC_ETS_2_CONSUMER

RegGrp

0x000B0160 - 0x000B017B

1.4.10.1 CONFIG

Reg.

0x000B0160

no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x20	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

1.4.10.2 STATUS

Reg.

0x000B0164

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled

					This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x0	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x0	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.10.3 CLAIM

Reg. 0x000B0168

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.10.4 CONTROL

Reg. 0x000B016C

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.10.5 INTR_STATE

Reg. 0x000B0170

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED

					• STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ • STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.10.6 INTR_ENABLE

0x000B0174

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL

1.4.10.7 INTR_TEST

0x000B0178

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.11 IPC_STE_2_ENTRY

0x000B0180 -
0x000B019F

rtl.hw_rp : true
display_name : ROT SECP to EXTP IPC Register
'Count' : 'IPC_STE_2_ENTRY' will repeat '8' times.

count	1	2	3	...	6	7	8

address	0xB0180	0xB0184	0xB0188	...	0xB0194	0xB0198	0xB019C

bits	name	s/w	h/w	default	description
31:0	DATA	rw	rw	0x0	SECP has read-write access to these registers when they are claimed (i.e. STATUS.CLAIM == 1). * When the STATUS.CLAIM is low, SECP writes are ignored EXTP has read-only access.

1.4.12 IPC_STE_2_PRODUCER

RegGrp 0x000B01A0 - 0x000B01BB

1.4.12.1 CONFIG

Reg. 0x000B01A0

no_reg_tests : true

bits	name	s/w	h/w	default	description
22:16	DEPTH	ro	ro	0x20	Number of EXTP to SECP bytes in Channel
8	CLAIM_DISABLE	rw	ro	0x1	0 - CLAIM operation functions for thread safe operation. 1 - CLAIM is forced high always. Thread safe operation is bypassed
6:0	THRESHOLD	rw	ro	0x4	When STATUS.AVAIL_BYTES matches or exceeds this Byte THRESHOLD, the STATUS.AVAIL flag will be set

1.4.12.2 STATUS

Reg. 0x000B01A4

no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
31	ADV_OVFL	r/w1c	rw	0x0	Set by hardware when firmware write CONTROL.ADVANCE with a value that would overflow or underrun the circular buffer hardware limits the advance to not exceed either full or empty state and sets this flag This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear rtl.hw_set : true
30	UNCLAIMED_READ	r/w1c	rw	0x0	Set by hardware when firmware reads consumer queue when STATUS.CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR This field will always be zero for Producer Write one to clear
29	UNCLAIMED	r/w1c	rw	0x0	Thread safe Queue operation Set by hardware when firmware writes to the ADVANCE register when CLAIM is not set This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
28	OUTOFBOUNDS	r/w1c	rw	0x0	Set by hardware when firmware writes to the queue beyond POINTER + AVAIL_BYTES mod DEPTH probably not useful when thread safe operation is enabled

					This status is enabled in INTR_ENABLE register, ORd with other enabled error status to create INTR_STATE.ERR Write one to clear
25	AVAIL	ro	wo	0x1	Set by hardware when the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD clear by writing INTR_STATE.AVAIL = 1
24	CLAIM	ro	rw	0x1	Thread safe Queue operation Set when firmware writes CLAIM.AVAIL_BYTES, CLAIM.POINTER that match STATUS.AVAIL_BYTES STATUS.POINTER values. Cleared by hardware when Firmware writes a non-zero value to CONTROL.ADVANCE value. This value is always high if the CONFIG.CLAIM_DISABLE is set rtl.hw_set : true rtl.hw_clear : true
14:8	AVAIL_BYTES	ro	rw	0x20	Number of bytes available in ETS/STE array. Producer: number of bytes available for writing Consumer: number of bytes available for reading AVAIL_BYTES[1:0] = 0 always
5:0	POINTER	ro	rw	0x0	Byte offset from the start of the ETS/STE array A available range is defined from Start of ETS/STE + STATUS.POINTER offset to Start of ETS/STE + (STATUS.POINTER + STATUS.AVAIL_BYTES) % CONFIG.DEPTH PRODUCER may write an available range CONSUMER may read an available range

1.4.12.3 CLAIM

Reg. 0x000B01A8

bits	name	s/w	h/w	default	description
5:0	POINTER	wo	ro	0x0	When CLAIM.POINTER is written and it matches STATUS.POINTER when STATUS.AVAIL == 1, STATUS.CLAIM is set

1.4.12.4 CONTROL

Reg. 0x000B01AC

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6:0	ADVANCE	rw	ro	0x0	Firmware writes Control ADVANCE with the number of bytes to advance the queue The value must be multiple of 4 - or results are not verified If firmware writes a value that exceeds IPC_DEPTH, the circular buffer will be either full for a producer or empty for a consumer and the STATUS.ADV_OVFL will be set

1.4.12.5 INTR_STATE

Reg. 0x000B01B0

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Hardware latches ERR when Any of the following status bits are active and enabled • STATUS.UNCLAIMED AND INTR_ENABLE.UNCLAIMED

					• STATUS.UNCLAIMED_READ AND INTR_ENABLE.UNCLAIMED_READ • STATUS.OUTOFBOUNDS AND INTR_ENABLE.OUTOFBOUNDS • STATUS.ADV_OVFL AND INTR_ENABLE.ADV_OVFL Write one to clear (after either clearing the error source in STATUS register or clearing the associated INTR_ENABLE). INTR_STATE.ERR, unlike INTR_STATE.AVAIL, **is** gated with INTR_ENABLE.ERR before driving the IPC interrupt. rtl.hw_set : true
0	AVAIL	r/w1c	rw	0x0	1: This bit is causing an IPC interrupt. 0: This bit is not causing an IPC interrupt. AVAIL is latched high when both • INTR_ENABLE.AVAIL is set AND • the bytes available in the queue match or exceed the value in CONFIG.THRESHOLD Write one to clear Note that this bit drives the IPC interrupt, ipc_int[1] directly. INTR_ENABLE.AVAIL == 0 inhibits INTR_STATE.AVAIL from latching high, but it does not gate ipc interrupt = INTR_STATE.AVAIL if INTR_STATE.AVAIL is already set. Note that for producers, after a reset, AVAIL will be latched high as soon as ***INTR_ENABLE*** .AVAIL is set, since the circular buffer is empty (available for writing) after a reset. rtl.hw_set : true

1.4.12.6 INTR_ENABLE

0x000B01B4

bits	name	s/w	h/w	default	description
31	ADV_OVFL	rw	ro	0x1	Enable ADV_OVFL to trigger INTR_STATE.ERR
30	UNCLAIMED_READ	rw	ro	0x1	Enable UNCLAIMED_READ to trigger INTR_STATE.ERR
29	UNCLAIMED	rw	ro	0x1	Enable UNCLAIMED to trigger INTR_STATE.ERR
28	OUTOFBOUNDS	rw	ro	0x1	Enable OUTOFBOUNDS to trigger INTR_STATE.ERR
1	ERR	rw	ro	0x1	Enable ERR to trigger ipc_int[0] interrupt
0	AVAIL	rw	ro	0x0	Enable AVAIL condition to latch INTR_STATE.AVAIL. Note that this ENABLE does not gate ipc_int[1] which is driven directly by INTR_STATE.AVAIL dontcompare : true

1.4.12.7 INTR_TEST

0x000B01B8

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	1: Sets INTR_STATE.ERR 0: no effect
0	AVAIL	rw	ro	0x0	1: Sets INTR_STATE.AVAIL 0: no effect

End RegGroup

1.4.13 SYS_STE_DOORBELL

0x000B0200 -
0x000B020B

no_reg_bit_bash_test : true
display_name : ROT SECP to EXTP Doorbells
'Count' : 'SYS_STE_DOORBELL' will repeat '3' times.

count	0	1	2
address	0xB0200	0xB0204	0xB0208

bits	name	s/w	h/w	default	description
0	DB	r/w1s	rw	0x0	SECP to EXTP Doorbell[i] is set. This bit is set by the SECP. The EXTP writes one to clear rtl.hw_clear : true

1.4.14 SYS_ETS_DOORBELL

0x000B0220 - 0x000B022B

no_reg_bit_bash_test : true
'Count' : 'SYS_ETS_DOORBELL' will repeat '3' times.

count	0	1	2
address	0xB0220	0xB0224	0xB0228

bits	name	s/w	h/w	default	description
0	DB	r/w1c	rw	0x0	EXTP to SECP Doorbell[i] is set. This bit is set by the EXTP. The SECP writes one to clear

1.4.15 SYS_ROT_STATUS_LOCK

0x000B0240

bits	name	s/w	h/w	default	description
0	ROT_STATUS_LOCK	rw	na	0x0	0: SYS_ROT_STATUS register may be written 1: SYS_ROT_STATUS register is locked for writes. It's value cannot change. 'lock' : 'ROT_STATUS_LOCK' is lock by 'ROT_STATUS_LOCK'.

1.4.16 SYS_LOCAL_STATUS_LOCK

0x000B0244

bits	name	s/w	h/w	default	description
0	LOCAL_STATUS_LOCK	rw	na	0x0	0: SYS_LOCAL_STATUS register may be written 1: SYS_LOCAL_STATUS register is locked for writes. It's value cannot change. 'lock' : 'LOCAL_STATUS_LOCK' is lock by 'LOCAL_STATUS_LOCK'.

1.4.17 SYS_ROT_STATUS

0x000B0248

bits	name	s/w	h/w	default	description
31:0	DATA	rw	ro	0x0	SECP firmware/romware may place boot progress in this register. If it is unable to finish the boot process, EXTP may read this register for failure analysis. EXTP has RO access to this register via EXTP_SYS_STATUS register. register is sticky - reset by rot_por_rstn hard_reset : false resetsignal : sys_por_rstn 'lock' : 'DATA' is lock by 'ROT_STATUS_LOCK'.

1.4.18 SYS_LOCAL_STATUS

0x000B024C

bits	name	s/w	h/w	default	description
31:0	DATA	rw	na	0x0	SECP firmware/romware may place private status in this register. For example, the register could track which memories have been initialzed. EXTP has no access to this register.

					register contents persist through rot_rstn - reset by rot_por_rstn hard_reset : false resetsignal : sys_por_rstn 'lock' : 'DATA' is lock by 'LOCAL_STATUS_LOCK'.

1.4.19 SYS_CLK_CTRL

Reg. 0x000B0260

bits	name	s/w	h/w	default	description
5	PERIPH_CLK_EN	rw	ro	0x0	1: enable clock to CS APB Peripherals 0: disable clock to CS APB Peripherals
4	RNG_CLK_EN	rw	ro	0x0	1: enable clock to RNG 0: disable clock to RNG
3	AES_CLK_EN	rw	ro	0x0	1: enable clock to AES 0: disable clock to AES
2	SHA_CLK_EN	rw	ro	0x0	1: enable clock to SHA 0: disable clock to SHA
1	ECA_CLK_EN	rw	ro	0x0	1: enable clock to ECA 0: disable clock to ECA
0	RSA_CLK_EN	rw	ro	0x0	1: enable clock to RSA 0: disable clock to RSA

1.4.20 SYS_RNG_CTRL

Reg. 0x000B0264

This register contains the control bit to reset the Crypto Assist Controller.

bits	name	s/w	h/w	default	description
1	RNG_ZEROIZE	rw	ro	0x0	Connected to Brouadcom RNG Controller if present Erases the RNG state. RNG synchronizes this signal and initiates zeroization on rising edge. Zeroization is actually a reset: all registers are cleared and self tests (of RNG core and ring oscillators) start. State information, seeds, keys (generated from ring oscillators output or provided by user) will be cleared. Ring oscillators during tests are disconnected and don't oscillate. The configuration registers are set to the default (reset) values
0	RNG_SHUTDOWN	rw	ro	0x1	Connected to Brouadcom RNG Controller if present allows shutting down the RNG to meet US regulation. A value of 1 on this signal disables the RNG clock. Reading of any RNG registers returns 0; writing to the registers has no effect. Firmware must clear this bit before using the TRNG_800_90AB dontcompare : true

1.4.21 SYS_OTP_CTRL

Reg. 0x000B0268

This register contains the control bit to soft reset the OTP Controller.

bits	name	s/w	h/w	default	description
1	OTP_ECC_DISABLE	rw	ro	0x0	Intended to Drive input, i_ecc_disable, on Broadcom OTP controller. ECC disable input. Disables ECC read when asserted. Wait until cmd_done is asserted before toggling this signal. Do not use the PROG_ECC and PROG_ECC_RP commands when i_ecc_disable is held high. Use the corresponding non-ECC commands

					instead - PROG and PROG_RP.
0	OTP_SOFT_RESET	rw	ro	0x0	1 - drive otp_soft_reset high to Vendor OTP Controller. This reset will reset OTPC but not OTP memory. otp_soft_reset does not trigger an autoload of HWControl bus. 0 - drive otp_soft_reset low (inactive) This bit must be low before using the OTP Controller. It is intended for recovery of unforseen hardware faults.

1.4.22 SYS_TESTMUX_CTRL

Reg. 0x000B026C

bits	name	s/w	h/w	default	description
1	TESTMUX_LOCK	rw	na	0x0	This bit once set cannot be cleared. When set, TESTMUX_ENABLE may not be changed. dontcompare : true 'lock' : 'TESTMUX_LOCK' is lock by 'TESTMUX_LOCK'.
0	TESTMUX_ENABLE	rw	ro	0x0	1: All Testmux selections are accessible by EXTP 0: Testmux selections above vector 1 are not accessible by EXTP (all values zero) Testmux selections 0 and 1 are always available (TESTMUXA/B_VERIF) if TESTMUX_LOCK is set, this field cannot change 'lock' : 'TESTMUX_ENABLE' is lock by 'TESTMUX_LOCK'.

1.4.23 SYS_CRYPTO_MEM_SEL

Reg. 0x000B0270

bits	name	s/w	h/w	default	description
0	ECA_SEL	rw	ro	0x1	1: CRYPTO memory is connected to ECA logic 0: CRYPTO memory is connected to RSA logic

1.4.24 SYS_NMI_VECTOR

Reg. 0x000B0274

bits	name	s/w	h/w	default	description
31:1	NMI_VECTOR	rw	ro	0x0	Connected directly to SECP NMI_VECTOR[31:1] to establish non-maskable interrupt vector

1.4.25 SYS_INTR_STATE

Reg. 0x000B0280

ECC Errors

bits	name	s/w	h/w	default	description
7	ECC_ERR_DB_KV_RAM	r/w1c	rw	0x0	This bit is set when KV_RAM Memory read results in a double bit ECC error in any of the four KV_RAM memory 32-bit lanes. Any bus master may cause the error. SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the SYS_ECC_STATUS_KV_RAM_TOP and SYS_ECC_STATUS_KV_RAM[i] registers. Firmware must set ECC_CTRL_KV_RAM[i].ERR_CLEAR to clear the appropriate double bit error state to enable future logging on that 32-bit lane. rtl.hw_set : true sticky : true
6	ECC_ERR_SB_KV_RAM	r/w1c	rw	0x0	This bit is set when KV_RAM Memory read results in a correctable ECC error in any of the four KV_RAM memory 32-bit lanes. Any bus master may cause the error.

					SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the SYS_ECC_STATUS_KV_RAM_TOP and SYS_ECC_STATUS_KV_RAM[i] registers. Firmware must set ECC_CTRL_KV_RAM[i].ERR_CLEAR to clear the appropriate single bit error state (including the single bit error counter) rtl.hw_set : true sticky : true
5	ECC_ERR_SB_SECP_RAM2	r/w1c	rw	0x0	This bit is set to 1 by hardware when any correctable ECC error occurs on a read from SECP_RAM2 from any bus master. SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the • SYS_ECC_STATUS_SECP_RAM2 • SYS_ECC_ADDR_SECP_RAM2 • SYS_ECC_DATA_SECP_RAM2 registers. Clearing this register does not clear the single bit error counter in SYS_ECC_STATUS_SECP_RAM2. rtl.hw_set : true sticky : true
4	ECC_ERR_DB_SECP_RAM2	r/w1c	rw	0x0	This bit is set to 1 by hardware when an uncorrectable ECC error occurs due to a non-SECP Instruction-Data read from SECP_RAM2. An access from any user besides USER_SECP_INST and USER_SECP_DATA can cause this status. A bus error occurs when SECP INSTR or DATA bus encounter an ECC DB error. SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the • SYS_ECC_STATUS_SECP_RAM2 • SYS_ECC_ADDR_SECP_RAM2 • SYS_ECC_DATA_SECP_RAM2 registers. Firmware must set ECC_CTRL_SECP_RAM2.ERR_CLEAR to clear the double bit error state to enable future logging rtl.hw_set : true sticky : true
3	ECC_ERR_SB_SECP_RAM1	r/w1c	rw	0x0	This bit is set to 1 by hardware when any correctable ECC error occurs on a read from SECP_RAM1 from any bus master. SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the • SYS_ECC_STATUS_SECP_RAM1 • SYS_ECC_ADDR_SECP_RAM1 • SYS_ECC_DATA_SECP_RAM1 registers. Clearing this register does not clear the single bit error counter in SYS_ECC_STATUS_SECP_RAM1. rtl.hw_set : true sticky : true
2	ECC_ERR_DB_SECP_RAM1	r/w1c	rw	0x0	This bit is set to 1 by hardware when an uncorrectable ECC error occurs due to a non-SECP Instruction-Data read from SECP_RAM1. An access from any user besides USER_SECP_INST and USER_SECP_DATA can cause

					this status. A bus error occurs when SECP INSTR or DATA bus encounter an ECC DB error. SECP clears this bit by writing a 1 to this bit position. Error logging details are found in the • SYS_ECC_STATUS_SECP_RAM1 • SYS_ECC_ADDR_SECP_RAM1 • SYS_ECC_DATA_SECP_RAM1 registers. Firmware must set ECC_CTRL_SECP_RAM1.ERR_CLEAR to clear the double bit error state to enable future logging rtl.hw_set : true sticky : true
1	ERR	r/w1c	rw	0x0	Unused, Will always be 0 rtl.hw_set : true sticky : true
0	ALERT	r/w1c	rw	0x0	Unused, Will always be 0 rtl.hw_set : true sticky : true

1.4.26 SYS_INTR_ENABLE

0x000B0284

SECP_RAM1 ECC Double Bit Error Interrupt Enable

bits	name	s/w	h/w	default	description
7	ECC_ERR_DB_KV_RAM	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_DB_KV_RAM to cause PIC interrupt
6	ECC_ERR_SB_KV_RAM	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_SB_KV_RAM to cause PIC interrupt
5	ECC_ERR_SB_SECP_RAM2	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_SB_SECP_RAM2 to cause PIC interrupt
4	ECC_ERR_DB_SECP_RAM2	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_DB_SECP_RAM2 to cause PIC interrupt
3	ECC_ERR_SB_SECP_RAM1	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_SB_SECP_RAM1 to cause PIC interrupt
2	ECC_ERR_DB_SECP_RAM1	rw	ro	0x0	1: Enable SYS_INTR_STATE.ECC_ERR_DB_SECP_RAM1 to cause PIC interrupt
1	ERR	rw	ro	0x0	Unused
0	ALERT	rw	ro	0x0	Unused

1.4.27 SYS_INTR_TEST

0x000B0288

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
7	ECC_ERR_DB_KV_RAM	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_DB_KV_RAM
6	ECC_ERR_SB_KV_RAM	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_SB_KV_RAM
5	ECC_ERR_SB_SECP_RAM2	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_SB_SECP_RAM2
4	ECC_ERR_DB_SECP_RAM2	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_DB_SECP_RAM2
3	ECC_ERR_SB_SECP_RAM1	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_SB_SECP_RAM1
2	ECC_ERR_DB_SECP_RAM1	rw	ro	0x0	1: Sets SYS_INTR_STATE.ECC_ERR_DB_SECP_RAM1
1	ERR	rw	ro	0x0	Unused
0	ALERT	rw	ro	0x0	Unused

1.4.28 SYS_ECC_CTRL_SECP_RAM1

0x000B0290

The ECC_CTRL register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
3	ERR_SB_INSERT	rw	ro	0x0	When set, a write to SECP_RAM causes correctable error by inverting DATA[0] of the SECP_RAM location
2	ERR_DB_INSERT	rw	ro	0x0	When set, a write to SECP_RAM causes uncorrectable error by inverting bits DATA[1:0] of the SECP_RAM location
1	ERR_CLEAR	wo	ro	0x0	When set, all pending ERR_DB, ERR_SB, and ERR_COUNT status are cleared. This bit is self-clearing 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ENABLE	rw	ro	0x0	1: enables ECC checking on SECP_RAM The SECP_RAM is protected with a 7-bit ECC per 32-bit word. Partial writes cause a RMW of the entire word and ECC.

1.4.29 SYS_ECC_STATUS_SECP_RAM1

0x000B0294

The ECC_STATUS register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
31	ERR_DB	ro	wo	0x0	Double bit ECC Error is pending caused by an SECP_RAM read with a double bit ECC error. No other errors will be logged until the field is cleared. Set ECC_CTRL_SECP_RAM.ERR_CLEAR to clear the field and allow future logging of ECC errors. Note that this bit only causes an interrupt to the PIC (SYS_INTR_STATE.DB_ERR) when the user master that performed the read to SECP_RAM is other than the SECP instruction or data user
30	ERR_SB	ro	wo	0x0	Single bit ECC Error occured. ERR_COUNT is non-zero. The field is cleared when ECC_CTRL_SECP_RAM.ERR_CLEAR is 1
3:0	ERR_COUNT	ro	wo	0x0	ERR_COUNT is incremented each time a single bit ECC error occurs. The count stops at 15. Set ECC_CTRL_SECP_RAM.ERR_CLEAR to clear the count

1.4.30 SYS_ECC_DATA_SECP_RAM1

0x000B0298

The ECC_LOG_DATA register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	Captured Data at ECC error location DATA is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR DATA is only valid when ERR_SB and/or ERR_DB are 1

1.4.31 SYS_ECC_ADDR_SECP_RAM1

0x000B029C

The ECC_LOG_ADDR register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>30:24</td><td>ECC</td><td>ro</td><td>wo</td><td>0x0</td><td>ECC is the captured syndrome of the single bit error
ECC is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR
ECC is only valid when ERR_SB and/or ERR_DB are 1</td></tr>
<tr><td>23:20</td><td>USER</td><td>ro</td><td>wo</td><td>0xF</td><td>USER is the Master Bus USER that triggered the ecc error
USER is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR
USER is only valid when ERR_SB and/or ERR_DB are 1
USER reflects USER_UNKNOWN when ERR_DB and ERR_SB are 0

enum:ROT_USER_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>USER_SECP_INST</td><td>0</td><td>SECP Instruction</td></tr>
<tr><td>USER_SECP_DATA</td><td>1</td><td>SECP Data</td></tr>
<tr><td>USER_SECP_DBG</td><td>2</td><td>SECP Debugger</td></tr>
<tr><td>USER_EXTP</td><td>3</td><td>EXTP</td></tr>
<tr><td>USER_KAM</td><td>4</td><td>KAM PMR Extend DMA</td></tr>
<tr><td>USER_DMA2</td><td>5</td><td>ROT DMA 2</td></tr>
<tr><td>USER_DMA3</td><td>6</td><td>ROT DMA 3</td></tr>
<tr><td>USER_DMA4</td><td>7</td><td>ROT DMA 4</td></tr>
<tr><td>USER_DMA5</td><td>8</td><td>ROT DMA 5</td></tr>
<tr><td>USER_NIC_AHBMTX</td><td>9</td><td>NIC TO AHB_MTX</td></tr>
<tr><td>USER_DMA0</td><td>10</td><td>DMA 0</td></tr>
<tr><td>USER_DMA1</td><td>11</td><td>DMA 1</td></tr>
<tr><td>USER_DAP_AXI</td><td>13</td><td>DAP AXI</td></tr>
<tr><td>USER_UNKNOWN</td><td>15</td><td>No User</td></tr>
</table>
encode : ROT_USER_e</td></tr>
<tr><td>18:0</td><td>ADDR</td><td>ro</td><td>wo</td><td>0x0</td><td>ADDR is the byte address of ECC error location
ADDR is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR
ADDR is only valid when ERR_SB and/or ERR_DB are 1</td></tr>
</table>

1.4.32 SYS_ECC_CTRL_SECP_RAM2

Reg. 0x000B02A0

The ECC_CTRL register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
3	ERR_SB_INSERT	rw	ro	0x0	When set, a write to SECP_RAM causes correctable error by inverting DATA[0] of the SECP_RAM location
2	ERR_DB_INSERT	rw	ro	0x0	When set, a write to SECP_RAM causes uncorrectable error by inverting bits DATA[1:0] of the SECP_RAM location
1	ERR_CLEAR	wo	ro	0x0	When set, all pending ERR_DB, ERR_SB, and ERR_COUNT status are cleared. This bit is self-clearing 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ENABLE	rw	ro	0x0	1: enables ECC checking on SECP_RAM

					The SECP_RAM is protected with a 7-bit ECC per 32-bit word. Partial writes cause a RMW of the entire word and ECC.

1.4.33 SYS_ECC_STATUS_SECP_RAM2

Reg. 0x000B02A4

The ECC_STATUS register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
31	ERR_DB	ro	wo	0x0	Double bit ECC Error is pending caused by an SECP_RAM read with a double bit ECC error. No other errors will be logged until the field is cleared. Set ECC_CTRL_SECP_RAM.ERR_CLEAR to clear the field and allow future logging of ECC errors. Note that this bit only causes an interrupt to the PIC (SYS_INTR_STATE.DB_ERR) when the user master that performed the read to SECP_RAM is other than the SECP instruction or data user
30	ERR_SB	ro	wo	0x0	Single bit ECC Error occured. ERR_COUNT is non-zero. The field is cleared when ECC_CTRL_SECP_RAM.ERR_CLEAR is 1
3:0	ERR_COUNT	ro	wo	0x0	ERR_COUNT is incremented each time a single bit ECC error occurs. The count stops at 15. Set ECC_CTRL_SECP_RAM.ERR_CLEAR to clear the count

1.4.34 SYS_ECC_DATA_SECP_RAM2

Reg. 0x000B02A8

The ECC_LOG_DATA register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	Captured Data at ECC error location DATA is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR DATA is only valid when ERR_SB and/or ERR_DB are 1

1.4.35 SYS_ECC_ADDR_SECP_RAM2

Reg. 0x000B02AC

The ECC_LOG_ADDR register definitions are the same for the two memories, SECP_RAM1 and SECP_RAM2. In the field descriptions substitute SECP_RAM1 or SECP_RAM2 in agreement with the instance name.

bits	name	s/w	h/w	default	description
30:24	ECC	ro	wo	0x0	ECC is the captured syndrome of the single bit error ECC is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR ECC is only valid when ERR_SB and/or ERR_DB are 1
23:20	USER	ro	wo	0xF	USER is the Master Bus USER that triggered the ecc error USER is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR USER is only valid when ERR_SB and/or ERR_DB are 1 USER reflects USER_UNKNOWN when ERR_DB and ERR_SB are 0

enum:ROT_USER_e

Name	Value	Description
USER_SECP_INST	0	SECP Instruction
USER_SECP_DATA	1	SECP Data
USER_SECP_DBG	2	SECP Debugger
USER_EXTP	3	EXTP
USER_KAM	4	KAM PMR Extend DMA
USER_DMA2	5	ROT DMA 2
USER_DMA3	6	ROT DMA 3
USER_DMA4	7	ROT DMA 4
USER_DMA5	8	ROT DMA 5
USER_NIC_AHBMTX	9	NIC TO AHB_MTX
USER_DMA0	10	DMA 0
USER_DMA1	11	DMA 1
USER_DAP_AXI	13	DAP AXI
USER_UNKNOWN	15	No User

encode : ROT_USER_e

18:0	ADDR	ro	wo	0x0	ADDR is the byte address of ECC error location ADDR is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_SECP_RAM.ERR_CLEAR ADDR is only valid when ERR_SB and/or ERR_DB are 1

1.4.36 SYS_ECC_CTRL_KV_RAM

0x000B02B0

KV_RAM Memory is organized as 256 rows of four 32-bit lanes. Each lane has independent ECC control and status.

bits	name	s/w	h/w	default	description
15:12	ERR_SB_INSERT	rw	ro	0x0	When ERR_SB_INSERT[i] is set, a KV_RAM write followed by read will cause correctable error on bit (i x 32) of the 128-bit KV_RAM location and will set ECC_STATUS_TOP_KV_RAM.ERR_SB[i]
11:8	ERR_DB_INSERT	rw	ro	0x0	When ERR_DB_INSERT[i] is set, a KV_RAM write followed by read will cause an uncorrectable error, flipping 2 consecutive bits starting at bit (i x 32) of the 128-bit KV_RAM location and will set ECC_STATUS_TOP_KV_RAM.ERR_DB[i]
7:4	ERR_CLEAR	wo	rw	0x0	When ERR_CLEAR[i] is set, any pending ERR_DB[i] and/or ERR_SB[i] will be cleared. The bits are self-clearing
0	ENABLE	rw	ro	0x0	1: enables ECC checking on the KV_RAM for the 32-bit lane. The KV_RAM is protected with a 7-bit ECC per 32-bit word. Byte and halfword writes are prohibited. Each RAM entry carries 128-bits of data with 28-bits of ECC.

1.4.37 SYS_ECC_STATUS_KV_RAM_TOP

0x000B02B4

bits	name	s/w	h/w	default	description
7:4	ERR_DB	ro	rw	0x0	ERR_DB[i] == 1 indicates that a double bit uncorrectable ECC error is detected in word[i]
3:0	ERR_SB	ro	rw	0x0	ERR_SB[i] == 1 indicates that a single bit correctable ECC error is detected in word[i]

1.4.38 SYS_ECC_STATUS_KV_RAM

0x000B02B8 - 0x000B02C7

KV_RAM Memory is organized as 256 rows of four 32-bit lanes. Each lane has independent ECC control and status.
'Count' : 'SYS_ECC_STATUS_KV_RAM' will repeat '4' times.

count	0	1	2	3
address	0xB02B8	0xB02BC	0xB02C0	0xB02C4

bits	name	s/w	h/w	default	description
31	ERR_DB	ro	wo	0x0	Double bit ECC Error is pending for the 32-bit lane. No other errors will be logged until the field is cleared. Set ECC_CTRL_KV_RAM.ERR_CLEAR to clear the field and allow future logging of ECC errors
30	ERR_SB	r/w1c	rw	0x0	Single bit ECC Error occured for the 32-bit lane Writing 1 to this field clears the status without affecting the ERR_COUNT. This field must be cleared before clearing the SYS_INTR_STATE.KV_RAM_ERR_SB The field is also cleared when ECC_CTRL_KV_RAM.ERR_CLEAR is 1 rtl.hw_set : true
27:24	ERR_COUNT	ro	wo	0x0	ERR_COUNT is incremented each time a single bit ECC error occurs for the 32-bit lane. The count stops at 15. Set ECC_CTRL_KV_RAM.ERR_CLEAR to clear the count. DMA0/1 reads with ARLEN> 0 can result in additional single bit error count. For instance, assume DMA0 read transaction with ARLEN=1, ARSIZE=32 from starting address 0x935E0. Since ARLEN=1, RoT hardware will issue two back to back 32bit reads from address locations 0x935E0 and 0x935E4. The kv_ram memory is configured as 256x156 (7+32+7+32+7+32+7+32) so, a memory read, returns 4 32bit words plus ecc. Kv_ram memory does not use the lower 4 bits of the address so it sees 0x93E50 and 0x93E54 as two memory reads to same address 0x935E. Now, if there was a single bit error in any of the 4 32bit words, error count will count twice because the same address was read two times
22:16	ECC	ro	wo	0x0	ECC is the captured syndrome of the single bit error for the 32-bit lane ECC is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_KV_RAM.ERR_CLEAR ECC is only valid when ERR_SB and/or ERR_DB are 1
15:12	USER	ro	wo	0xF	USER is the Master Bus USER that triggered the ecc error for the 32-bit lane USER is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_KV_RAM.ERR_CLEAR USER is only valid when ERR_SB and/or ERR_DB are 1 USER reflects USER_UNKNOWN when ERR_DB and ERR_SB are 0

enum:ROT_USER_e

Name	Value	Description
USER_SECP_INST	0	SECP Instruction
USER_SECP_DATA	1	SECP Data
USER_SECP_DBG	2	SECP Debugger

USER_EXTP	3	EXTP
USER_KAM	4	KAM PMR Extend DMA
USER_DMA2	5	ROT DMA 2
USER_DMA3	6	ROT DMA 3
USER_DMA4	7	ROT DMA 4
USER_DMA5	8	ROT DMA 5
USER_NIC_AHBMTX	9	NIC TO AHB_MTX
USER_DMA0	10	DMA 0
USER_DMA1	11	DMA 1
USER_DAP_AXI	13	DAP AXI
USER_UNKNOWN	15	No User

encode : ROT_USER_e

11:0	ADDR	ro	wo	0x0	ADDR is the byte address of ECC error location for the 32-bit lane ADDR is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_KV_RAM.ERR_CLEAR ADDR is only valid when ERR_SB and/or ERR_DB are 1

1.4.39 SYS_ECC_DATA_KV_RAM

0x000B02C8 - 0x000B02D7

'Count' : 'SYS_ECC_DATA_KV_RAM' will repeat '4' times.

count	0	1	2	3
address	0xB02C8	0xB02CC	0xB02D0	0xB02D4

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	Captured Data at ECC error location DATA is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware sets ECC_CTRL_KV_RAM.ERR_CLEAR DATA is only valid when ERR_SB and/or ERR_DB are 1

1.4.40 SYS_MEM_CTRL_ROM

0x000B02D8

ROM Memory Control

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high. TBD
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit

1.4.41 SYS_MEM_CTRL_SECP_RAM1

0x000B02DC

SECP_RAM1 Memory Control

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high.
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit
1:0	POFF	rw	ro	0x0	Peripheral off?

1.4.42 SYS_MEM_CTRL_SECP_RAM2

Reg. 0x000B02E0

These controls are only connected to memories for CSSD host SOC.
Refer to the https://wdc.box.com/s/jrq7a86cxv5v40iop9ozhrjr9jvonujp]Architecture Spec
for used controls per SoC.

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high. TBD
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit

1.4.43 SYS_MEM_CTRL_MAA

Reg. 0x000B02E4

MAA Memory Control

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high. TBD
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit

1.4.44 SYS_MEM_CTRL_RSA

Reg. 0x000B02E8

RSA Memory Control (CS Only)

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high. TBD
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit

1.4.45 SYS_MEM_CTRL_PMR

Reg. 0x000B02EC

PCR Memory Control

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high.
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit
1:0	POFF	rw	ro	0x0	Peripheral off?

1.4.46 SYS_MEM_CTRL_KV_RAM

Reg. 0x000B02F0

KV_RAM Memory Control

bits	name	s/w	h/w	default	description
11:9	BC	rw	ro	0x2	Biasing bypass input pin
8	RMEN	rw	ro	0x0	Read Margin Enable
7:4	RM	rw	ro	0x3	Read Margin
3	TEST_RNM	rw	ro	0x0	Memory will go in Idle state and bit-lines are pre-charged when this ping is high. TBD
2	TEST1	rw	ro	0x0	Test pin to bypass self-timed circuit

1.4.47 SYS_PM_FRAME_TABLE

RegGrp 0x000B0340 - 0x000B034F

The Frame Configuration Table (Frame Table) is part of the Page Management Static configuration. Each Frame Table Entry specifies
the start address and page size for a frame-array in SECP_RAM consisting of 32 frames.
PM hardware indexes the array with PM_TABLE[pm_index].**FRAME_TABLE_INDEX** for virtual to physical address mapping.
'stride' : '8' Specifies the address stride when instantiating an array of components
'Count' : 'SYS_PM_FRAME_TABLE' will repeat '2' times.

count	0	1
address	0xB0340	0xB0348

1.4.47.1 PHYSICAL_ADDR

Reg. 0x000B0340

bits	name	s/w	h/w	default	description
31:12	ADDR_HI	rw	ro	0x30	Physical base address of a frame array to which pages are stored. The PM table entries contain FRAME_TABLE_INDEX, indexing this FRAME_TABLE for the virtual to physical address calculation. The Frame array must start on a multiple of the page size. In other words, if 8KB pages are used, firmware must set ADDR_HI[0] (PHYSICAL_ADDR[12]) = 0. 'lock' : 'ADDR_HI' is lock by 'PM_ENABLE'.
11:0	ADDR_LO	ro	ro	0x0	always 0

1.4.47.2 PAGE_SIZE

Reg. 0x000B0344

Part of the PM Static configuration

bits	name	s/w	h/w	default	description
1:0	PAGE_SIZE	rw	ro	0x1	Encoded page size - used by hardware for indexing into the physical frame array(s) bytesPerPage = 2 ^^ (10 + PAGE_SIZE) 'lock' : 'PAGE_SIZE' is lock by 'PM_ENABLE'.

End RegGroup

1.4.48 SYS_PM_CTRL

Reg. 0x000B0350

Part of the PM Static configuration

bits	name	s/w	h/w	default	description
0	PM_ENABLE	rw	ro	0x0	1: enables Page Manager feature and locks PM configuration registers. 0: no virtual mapping. See memory maps for more information.

1.4.49 SYS_PM_MISS_STATUS

Reg. 0x000B0360

PM related status

bits	name	s/w	h/w	default	description
2	MISS_DBG	r/w1c	wo	0x0	1: When PM_ENABLE == 1, the field is set by hardware when the SECP DEBUG user attempts to access Overlay memory space and no PM Table Page Address

					matches the address for the access. write one to clear sticky : true
1	MISS_LSU	r/w1c	wo	0x0	1: When PM_ENABLE == 1, the field is set by hardware when the SECP_DATA user attempts to access Overlay memory space and no PM Table Page Address matches the address for the access. write one to clear sticky : true
0	MISS_IFU	r/w1c	wo	0x0	1: When PM_ENABLE == 1, the field is set by hardware when the SECP INSTR user attempts to access Overlay memory space and no PM Table Page Address matches the address for the access. write one to clear sticky : true

1.4.50 SYS_PM_DIRTY_STATUS

Reg. 0x000B0364

bits	name	s/w	h/w	default	description
31:0	DIRTY	r/w1c	rw	0x0	When PM_ENABLE == 1, hardware sets this DIRTY[pageTableIndex] when a SECP write occurs to an Overlay memory address that is within the page spacified by PM_TABLE[pageTableIndex]. write one to clear sticky : true

1.4.51 SYS_PM_MRU_RESET

Reg. 0x000B0370

bits	name	s/w	h/w	default	description
0	MRU_QEUUE_RESET	rw	rw	0x0	Restore default reset values to MRU Queue when SYS_PM_CTRL.PM_ENABLE == 0. Self-clearing dontcompare : true sticky : true

1.4.52 SYS_PM_MRU_ATTRIBUTE

Reg. 0x000B0374

bits	name	s/w	h/w	default	description

31:0	ATTRIBUTE	ro	wo	0x0	Hardware orders the 32
					PAGE_TABLE[i].CONTROL.ATTRUTE bits into a single 32-bit register based on the page_table index in the MRU Queue.
					Firmware may use this organization of ATTRIBUTEs to analyze the MRU Queue
					<pre>
					for (int i = 0; i < 32 i++) {
					int page_table_index = PM_MRU[i];
					ATTRIBUTE[i] =
					PAGE_TABLE[page_table_index].CONTROL.ATTRIBUTE;
					}
					</pre>

1.4.53 SYS_PM_MRU_RESIDENT

Reg. 0x000B0378

bits	name	s/w	h/w	default	description
31:0	RESIDENT	ro	wo	0x0	Hardware orders the 32 PAGE_TABLE[i].CONTROL.ATTRUTE bits into a single

					32-bit register based on the page_table index in the MRU Queue. Firmware may use this organization of RESIDENTs to analyze the MRU Queue <pre> for *int i = 0; i < 32 i++) begin int page_table_index = PM_MRU[i]; RESIDENT[i] = PAGE_TABLE[page_table_index].CONTROL.RESIDENT; end </pre>

1.4.54 SYS_PM_MRU

Reg. 0x000B0380 - 0x000B03FF

This register array is a copy of the MRU QUEUE used by hardware.
Firmware cannot write the queue directly but may initialize the queue by writing 1 to the SYS_PM_MRU_RESET register when PM_ENABLE == 0.
The queue will always contain 32 unique PM_TABLE indices.
When PM hardware finds a match on PM_TABLE[pm_table_index].PAGE_ADDR, hardware will

- delete the SYS_PM_MRU entry that contains *pm_table_index*
- push *pm_table_index* into SECP_SYS_PM_MRU[0]
- shift existing entries towards the deleted entry

The resulting MRU queue has most recently used items near the lower indices and least recently used indices at the higher indices; i.e., the least recently used page is in SECP_SYS_PM_MRU[31].
If only 16 PM entries are in use and resident, the least recently used Resident entry will eventually reside at SECP_SYS_PM_MRU[15].
no_reg_bit_bash_test : true
'Count' : 'SYS_PM_MRU' will repeat '32' times.

count	1	2	3	...	30	31	32
address	0xB0380	0xB0384	0xB0388	...	0xB03F4	0xB03F8	0xB03FC

bits	name	s/w	h/w	default	description
4:0	PM_TABLE_INDEX	ro	rw	0x0	An index into the PM table

1.4.55 SYS_PM_TABLE

RegGrp 0x000B0400 - 0x000B04FF

The PM Table comprises 32 entries. Each entry has

- Resident bit
- Attribute bit
- Page Address
- Frame Table Index (1-bit)
- Frame index (5-bit)

'stride' : '8' Specifies the address stride when instantiating an array of components
'Count' : 'SYS_PM_TABLE' will repeat '32' times.

count	1	2	3	...	30	31	32
address	0xB0400	0xB0408	0xB0410	...	0xB04E8	0xB04F0	0xB04F8

1.4.55.1 CONTROL

Reg. 0x000B0400

bits	name	s/w	h/w	default	description
31	RESIDENT	rw	ro	0x0	0: Hardware ignores this entry in its Page match search 1: Hardware includes this entry in its Page match search when the SECP is in the OVERLAY range

16	ATTRIBUTE	rw	ro	0x0	Bit may be encoded by firmware to indicate whether the the Page is read-only or read-write. This bit will be presented in most recenlty used order in SYS_MRU_ATTRIBUTE register. Otherwise this bit is unused by hardware.
8	FRAME_TABLE_INDEX	rw	ro	0x0	0: Base offset frame array comes from SECP_SYS_FRAME_TABLE[0] 1: Base offset frame array comes from SECP_SYS_FRAME_TABLE[1]

4:0	FRAME_INDEX	rw	ro	0x0	The index of the frame of size and address indicated by FRAME_TABLE[FRAME_TABLE_INDEX]
					`page_size =`
					`get_page_size(SECP_SYS_FRAME_TABLE[ FRAME_TABLE_IN`
					`Physical Address =`
					`SECP_SYS_FRAME_TABLE[ FRAME_TABLE_INDEX].PHYSICAL_`
					`+ FRAME_INDEX * page_size;`

1.4.55.2 PAGE_ADDR

0x000B0404

bits	name	s/w	h/w	default	description
31:28	UPPER	ro	ro	0x0	Upper 4-bits of virtual page address always zero
27:10	HI	rw	ro	0x0	Specifies the virtual address of a the start of a virtual page. Firmware configures page size to either 1KB, 2KB, 4KB or 8KB. When the page size is configured to 1KB, hardware tries to match all PAGE_ADDR[27:10] When the page size is configured to 2KB, hardware ignores PAGE_ADDR[11:10] . When the page size is configured to 8KB, hardware ignores PAGE_ADDR[12:10].
9:0	LO	ro	ro	0x0	Lower 10 bits of virtual page address always zero - never compared

End RegGroup

1.4.56 SYS_DMA_MPU

0x000B0500 - 0x000B057F

'Count' : 'SYS_DMA_MPU' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xB0500	0xB0510	0xB0520	...	0xB0550	0xB0560	0xB0570

1.4.56.1 RANGE_START

0x000B0500

bits	name	s/w	h/w	default	description
31:12	ADDR_HI	rw	ro	0x0	Start address for MPU range entry. The defined range is from DMA_MPU_START to DMA_MPU_END 'lock' : 'ADDR_HI' is lock by 'DMA_MPU_LOCK'.
11:0	ADDR_LO	ro	ro	0x0	4KB granularity - lower address bits zero

1.4.56.2 RANGE_END

0x000B0504

bits	name	s/w	h/w	default	description
31:12	ADDR_HI	rw	ro	0x0	End address for MPU range entry - The defined range is from DMA_MPU_START to DMA_MPU_END 'lock' : 'ADDR_HI' is lock by 'DMA_MPU_LOCK'.
11:0	ADDR_LO	ro	ro	0xFFF	4KB granularity - lower address bits all ones

1.4.56.3 WRITE_PROTECT

0x000B0508

bits	name	s/w	h/w	default	description
0	WP	rw	ro	0x0	1: The Address Range specified in this DMA MPU entry cannot be written by DMA 0: The Address Range specified in this DMA MPU does not inhibit DMA writes 'lock' : 'WP' is lock by 'DMA_MPU_LOCK'.

1.4.56.4 READ_PROTECT

0x000B050C

bits	name	s/w	h/w	default	description
0	RP	rw	ro	0x0	1: The Address Range specified in this DMA MPU entry cannot be read by ROT DMA 0: The Address Range specified in this DMA MPU does not inhibit ROT DMA reads 'lock' : 'RP' is lock by 'DMA_MPU_LOCK'.

End RegGroup

1.4.57 SYS_DMA_MPU_LOCK

0x000B0580

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
0	DMA_MPU_LOCK	rw	ro	0x0	0: DMA MPU Table may be modified 1: DMA MPU Table cannot be modified Once set, this bit is latched high until a por_rstn or warm_rstn 'lock' : 'DMA_MPU_LOCK' is lock by 'DMA_MPU_LOCK'.

1.4.58 SYS_EXTP_PERMIT

0x000B0600

Permission for EXTP access to RoT slaves. These permissions are controlled by

1. SoC Global Access Rules
2. These fields (controlled by SECP only) if permitted by SoC Global Access Rules

SECP will not be able to set a field that is disabled by the Global SoC Permission Rules.
SECP will not be able to change any field if the SECP SYS_EXTP_PERMIT_LOCK is set.
For most RoT Instances, these fields are fixed by Global Access Rules. For CS Instance, the DMA_0 is writeable by SECP. Future instantiations may have more access permissions. TBD.

bits	name	s/w	h/w	default	description
19	GPIO	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'GPIO' is lock by 'PERMIT_LOCK'.
18	I2C	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'I2C' is lock by 'PERMIT_LOCK'.
17	SPI	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'SPI' is lock by 'PERMIT_LOCK'.
16	UART	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'UART' is lock by 'PERMIT_LOCK'.
15	DMA_5	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'DMA_5' is lock by 'PERMIT_LOCK'.
14	DMA_4	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed

					'lock' : 'DMA_4' is lock by 'PERMIT_LOCK'.
13	DMA_3	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'DMA_3' is lock by 'PERMIT_LOCK'.
12	DMA_2	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'DMA_2' is lock by 'PERMIT_LOCK'.
11	DMA_1	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'DMA_1' is lock by 'PERMIT_LOCK'.
10	DMA_0	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed For CS SoC Only, This field will be set to 1 on rot_rstn. SECP has R/W access this permission bit for EXPT DMA 0 access (until locked by EXTP_PERMIT_LOCK). 'lock' : 'DMA_0' is lock by 'PERMIT_LOCK'.
9	RNG	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'RNG' is lock by 'PERMIT_LOCK'.
8	RSA	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'RSA' is lock by 'PERMIT_LOCK'.
7	MAA	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'MAA' is lock by 'PERMIT_LOCK'.
6	ECA	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'ECA' is lock by 'PERMIT_LOCK'.
5	AES_1	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'AES_1' is lock by 'PERMIT_LOCK'.
4	AES_0	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'AES_0' is lock by 'PERMIT_LOCK'.
3	SHA	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'SHA' is lock by 'PERMIT_LOCK'.
2	KAM	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'KAM' is lock by 'PERMIT_LOCK'.
1	OTP	rw	rw	0x0	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'OTP' is lock by 'PERMIT_LOCK'.
0	SYS_EXTP	rw	rw	0x1	0: no permission to access from EXTP 1: access from EXTP allowed 'lock' : 'SYS_EXTP' is lock by 'PERMIT_LOCK'.

1.4.59 SYS_EXTP_PERMIT_LOCK

Reg. 0x000B0604

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
0	PERMIT_LOCK	rw	rw	0x1	0: OK to alter changable EXTP_PERMIT fields 1: All EXTP_PERMIT fields are as well as PERMIT_LOCK are unwritable) CS SoCs will differ in the advertised reset value. CS SoCs will read zero from PERMIT_LOCK after rot_rstn. dontcompare : true 'lock' : 'PERMIT_LOCK' is lock by 'PERMIT_LOCK'.

1.4.60 SYS_SCRATCH_2

Reg. 0x000B0704

bits	name	s/w	h/w	default	description
31:0	DATA	rw	rw	0x0	test register for firmware use only

					The register is only reset by rot_por_rstn. EXTP also has R/W access to this register

End RegGroup

1.5 ROT_OTP

RegGrp 0x000C3000 - 0x000C3203

decode_size : 0x00001000
chapter : 1.8, Security Subsystem, ROT Vendor OTP Access
blockgroup : SECUREPROCESSOR
revision : revision: cdb7dc6

1.5.1 OTP_OTPC

RegGrp 0x000C3000 - 0x000C30FF

1.5.1.1 OTP_OTPC_CONTROL

Reg. 0x000C3000

rtl.reg_enb : false
'OTP_OTPC_CONTROL' is an external.

bits	name	s/w	h/w	default	description
1	OTP_DO_NOT_USE	rw	ro	0x0	This field is reserved for future use. It should not be set since its use is in hardware is undefined.
0	OTP_CPU_MODE_EN	rw	ro	0x0	Write a 1 to this to be able to access OTP through the APB interface. If this is 0, OTP is only accessable through JTAG (by default) dontcompare : true

1.5.1.2 OTP_OTPC_ADDRESS

Reg. 0x000C3004

rtl.reg_enb : false
'OTP_OTPC_ADDRESS' is an external.

bits	name	s/w	h/w	default	description
9:0	OTP_ADDR	rw	ro	0x0	The program address input bus is used to select a word out of OTP array

1.5.1.3 OTP_OTPC_CONTROL0

Reg. 0x000C3008

rtl.reg_enb : false
'OTP_OTPC_CONTROL0' is an external.

bits	name	s/w	h/w	default	description
5:1	OTP_CMD	rw	ro	0x0	This field specifies the OTP command
0	START	wo	ro	0x0	Start bit to tell OTP controller to send command to OTP controller. APB should set this bit after it has set OTP_CMD and other registers. This bit is self clearing. dontcompare : true

1.5.1.4 OTP_OTPC_STATUS0

Reg. 0x000C300C

rtl.reg_enb : false
'OTP_OTPC_STATUS0' is an external.
no_reg_tests : true

bits	name	s/w	h/w	default	description

23	OTP_ECC_DED_FLAG_STATUS	ro	wo	0x0	DED flag status. This bit is set when ECC with DED is enabled and double error is detected.
22	OTP_ECC_SEC_FLAG_STATUS	ro	wo	0x0	SEC flag status. This bit is set when ECC with SEC is enabled and double error is detected.
21	OTP_ECC_CORRECTION_STATUS	ro	wo	0x0	ECC correction status. This bit is valid when ECC is enabled and indicates whether the ECC correction is enabled.
20	OTP_PRESCREEN_FAIL	ro	wo	0x0	Prescreen fail. This bit is set during PRESCREEN command for any failure
19	OTP_RWP_ECC_DED_STATUS	ro	wo	0x0	Applicable for AUTOREPAIR Feature
18	OTP_RWP_ECC_SEC_STATUS	ro	wo	0x0	Applicable for AUTOREPAIR Feature
17	LOAD_RF_DONE	ro	wo	0x0	Applicable for AUTOREPAIR Feature
16	OTP_MAX_RWP	ro	wo	0x0	Applicable for AUTOREPAIR Feature
15	OTP_MAX_RW	ro	wo	0x0	Applicable for AUTOREPAIR Feature
14	OTP_PRGM_WD_RP_FAIL	ro	wo	0x0	Applicable for AUTOREPAIR Feature
13	OTP_PROG_EN	ro	wo	0x1	PROG enable bit. By default this is set to enable PROG command.
12	OTP_PROG_BLOCK_CMD	ro	wo	0x0	Blocked PROG related commands for Secure space. Prog command blocked for secure space. Only PROG_LOCK can be used to program.
11	OTP_PROG_SCREEN_FAIL	ro	wo	0x0	Program screen failure. This bit is set when screening fails for word programming.
10	OTP_PROG_WORD_FAIL	ro	wo	0x0	Program word failure. This bit is set when Programming fails for a bit during word Program. This bit is set if PROGRAM command is issued when PROGOK is not enabled
9	OTP_INVALID_ADDR	ro	wo	0x0	Invalid address entered. This bit is set when Locked address is accessed by program related commands or when address is out of range.
8	OTP_DEBUG_ENABLE	ro	wo	0x0	Debug mode register. This bit is set using ctrl_wr command and indicates the debug mode option. Debug mode provides the direct interaction with OTP memory. Note: When TIECELL UNLOCK DISABLE bit is blown, this bit is not applicable.
7	OTP_MST_FSM_ERROR	ro	wo	0x0	An illegal state has executed. This bit is set to '0' in idle state, otherwise '1' in all other states. Note: This bit indicates whether if state machine is stuck during command execution
6	OTP_DEBUG_MODE_SET	ro	wo	0x0	This bit is set when ctrl_wr_cmd is issued. (N/A for APB)
5	OTP_REFOK	ro	wo	0x0	OTP RefOK signal
4	OTP_CMD_FAIL	ro	wo	0x0	Command Failure. 1) This bit is set when Locked address is accessed using program related commands: a) PROG command access lock rows ('d11 and '12). b) PROG_LOCK command access addresses which are not defined under LOCK addresses. 2) This bit is also set when READ_TEST command is executed with incorrect parameters. (N/A for APB)
3	OTP_FDONE	ro	wo	0x1	FDone. (Same as VENDOR_READY) This signal is set when fout bits (VENDOR_BITS) are loaded if autoload is enabled.
2	OTP_PROGOK	ro	wo	0x0	OTP ProgOK signal. This signal is set when PROG ENABLE sequence is issued correctly
1	OTP_CMD_DONE	ro	wo	0x1	Command Done. This signal indicates the completion of the command
0	OTP_DATA_VALID	ro	wo	0x0	Data Valid. Not Applicable for APB. Note: For single READ, use cmd_done for valid data

1.5.1.5 OTP_OTPC_STATUS1

Reg. 0x000C3010

rtl.reg_enb : false
'OTP_OTPC_STATUS1' is an external.

bits	name	s/w	h/w	default	description

0	COMMAND_DONE	r/w1c	ro	0x0	This bit is set when the state machine has returned to IDLE. Hardware sets it to one when the command completes. Write 1 to clear. APB should clear this bit before sending a new command.

1.5.1.6 OTP_OTPC_WRITE

Reg. 0x000C3040 - 0x000C3047

rtl.reg_enb : false
'OTP_OTPC_WRITE' is an external.
'Count' : 'OTP_OTPC_WRITE' will repeat '2' times.

count	0	1
address	0xC3040	0xC3044

bits	name	s/w	h/w	default	description
31:0	DATA	rw	ro	0x0	DATA[0] Bits [31:0] of data to be written/programmed into OTP DATA[1] Bits [63:32] of data to be written/programmed into OTP

1.5.1.7 OTP_OTPC_READ

Reg. 0x000C3080 - 0x000C3087

rtl.reg_enb : false
'OTP_OTPC_READ' is an external.
'Count' : 'OTP_OTPC_READ' will repeat '2' times.

count	0	1
address	0xC3080	0xC3084

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	DATA[0]: Bits [31:0] read data returned from command. DATA[1]: Bits [63:32] read data returned from command.

1.5.1.8 OTP_OTPC_UNUSED

Reg. 0x000C3088 - 0x000C30FF

rtl.reg_enb : false
'OTP_OTPC_UNUSED' is an external.
'Count' : 'OTP_OTPC_UNUSED' will repeat '30' times.

count	1	2	3	...	28	29	30
address	0xC3088	0xC308C	0xC3090	...	0xC30F4	0xC30F8	0xC30FC

bits	name	s/w	h/w	default	description
31:0	UNUSED	r/w1c	rw	0x0	

End RegGroup

1.5.2 OTP_EFC

RegGrp 0x000C3100 - 0x000C31FF

1.5.2.1 OTP_EFC_CONTROL

Reg. 0x000C3100

rtl.reg_enb : false
'OTP_EFC_CONTROL' is an external.
no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
6	MR	rw	ro	0x0	
5	SWR	rw	rw	0x0	
4	IRQ_MODE	rw	ro	0x0	

3	IRQ_EN	rw	ro	0x0	
2	BYPASS_MODE	rw	ro	0x0	
1	START_RD	rw	rw	0x0	
0	START_PGM	rw	rw	0x0	

1.5.2.2 OTP_EFC_BYPASS_CONTROL

Reg. 0x000C3104

rtl.reg_enb : false
'OTP_EFC_BYPASS_CONTROL' is an external.
no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
19	PGEN	rw	ro	0x0	
18	LOAD	rw	ro	0x1	
17	STB	rw	ro	0x0	
16	CSB	rw	ro	0x1	
9:0	ADDRESS	rw	ro	0x0	

1.5.2.3 OTP_EFC_STATUS

Reg. 0x000C3108

rtl.reg_enb : false
'OTP_EFC_STATUS' is an external.

bits	name	s/w	h/w	default	description
23:16	CALCULATED_CRC_DATA	ro	ro	0x0	
8	CRC_MATCH	ro	ro	0x0	
4	IRQ_ST_RD_DONE	r/w1c	ro	0x0	
3	IRQ_ST_PGM_DONE	r/w1c	ro	0x0	
2	RD_FSM_BUSY	ro	ro	0x0	
1	PGM_FSM_BUSY	ro	ro	0x0	
0	FSM_BUSY	ro	ro	0x0	

1.5.2.4 OTP_EFC_POWER_CONTROLS

Reg. 0x000C310C

rtl.reg_enb : false
dontcompare : true
'OTP_EFC_POWER_CONTROLS' is an external.

bits	name	s/w	h/w	default	description
1	PD	rw	ro	0x1	
0	PS	rw	ro	0x0	

1.5.2.5 OTP_EFC_ACCESS_TIMERS

Reg. 0x000C3110

rtl.reg_enb : false
'OTP_EFC_ACCESS_TIMERS' is an external.

bits	name	s/w	h/w	default	description
31:29	RFU	rw	ro	0x0	
25:16	HOLD_STROBE_1_TIMING	rw	ro	0x0	
15:8	SETUP_STROBE_0_TIMING	rw	ro	0x0	
3:0	SETUP_HOLD_TIMING	rw	ro	0x0	

1.5.2.6 OTP_EFC_ADDRESS

Reg. 0x000C3114

rtl.reg_enb : false
'OTP_EFC_ADDRESS' is an external.

bits	name	s/w	h/w	default	description
4:0	EFC_DW_ADDRESS	rw	ro	0x0	

1.5.2.7 OTP_EFC_PGM_DATA

Reg. 0x000C3118

rtl.reg_enb : false
'OTP_EFC_PGM_DATA' is an external.

bits	name	s/w	h/w	default	description
31:0	PGM_DATA	rw	ro	0x0	

1.5.2.8 OTP_EFC_RD_DATA

Reg. 0x000C311C

rtl.reg_enb : false
'OTP_EFC_RD_DATA' is an external.

bits	name	s/w	h/w	default	description
31:0	RD_DATA	ro	ro	0x0	

1.5.2.9 OTP_EFC_WR_DELAY

Reg. 0x000C3120

rtl.reg_enb : false
'OTP_EFC_WR_DELAY' is an external.

bits	name	s/w	h/w	default	description
8:6	WRITE_STROBE_DELAY	rw	ro	0x2	
5:3	DELAY_AFTER_PROG	rw	ro	0x4	
2:0	ADDR_CHANGE_DELAY	rw	ro	0x4	

1.5.2.10 OTP_EFC_UNUSED

Reg. 0x000C3124 -
0x000C31FF

rtl.reg_enb : false
'OTP_EFC_UNUSED' is an external.
'Count' : 'OTP_EFC_UNUSED' will repeat '55' times.

count	1	2	3	...	53	54	55
address	0xC3124	0xC3128	0xC312C	...	0xC31F4	0xC31F8	0xC31FC

bits	name	s/w	h/w	default	description
31:0	UNUSED	r/w1c	rw	0x0	

End RegGroup

1.5.3 OTP_LOCK

Reg. 0x000C3200

This register access is only forwarded to APB when the ROT_OTP_SHARE feature is enabled.
When ROT_OTP_SHARE = 1, RoT must arbitrate for OTP Controller access.

1. Write 1 to OTP_LOCK.OTP_REQ

2. Read OTP_LOCK.OTP_GRANT_STATUS till it is 1

3. Access OTP Controller

4. Write 0 to OTP_LOCK.OTP_REQ

When ROT_OTP_SHARE = 0, this register has no effect on OTP Controller operation.
rtl.reg_enb : false
'OTP_LOCK' is an external.

bits	name	s/w	h/w	default	description
1	OTP_GRANT_STATUS	ro	rw	0x0	1: OK to access OTP Controller 0: Not OK to access OTP Controller if OTP_SHARE feature is enabled

					OTP_SHARE == 0: firmware has control of its dedictated OTP APB bus regardless of this field dontcompare : true
0	OTP_REQ	rw	ro	0x0	Write 1 to request exclusive control of the OTP APB Bus Write 0 to release the request (and any pending grant) OTP_SHARE == 1: This register is mirrored external to RoT via APB.

End RegGroup

1.6 ROT_KAM

RegGrp

0x000C4000 -
0x000C4703

decode_size : 0x00001000
chapter : 1.7, Security Subsystem, ROT Vendor OTP Access
blockgroup : SECUREPROCESSOR
revision : revision: cdb7dc6

1.6.1 KAM_PMR_LIST

RegGrp

0x000C4000 -
0x000C41FF

'Count' : 'KAM_PMR_LIST' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xC4000	0xC4040	0xC4080	...	0xC4140	0xC4180	0xC41C0

1.6.1.1 KAM_PMR_ENTRY

Reg.

0x000C4000 -
0x000C403F

Platform Configuration Registers (PMR). Each PMR slot contains 16 32-bit PMR values.
4,8,12, or 32 DATA regs comprise a PMR.
rtl.reg_enb : false
'KAM_PMR_ENTRY' is an external.
no_reg_bit_bash_test : true
reg_wprot : priv
'Count' : 'KAM_PMR_ENTRY' will repeat '16' times.

count	1	2	3	...	14	15	16
address	0xC4000	0xC4004	0xC4008	...	0xC4034	0xC4038	0xC403C

bits	name	s/w	h/w	default	description
31:0	DATA	rw	wo	0x0	PMR data: Data are organized as most significant 32-bits at index 0 Each PMR consists of 4,8,12,or 16 32-bit DATA entries FW cannot write the PMR directly. Use PMR Extend function to update PMRs. The PMR Extend function writes using this address (FW access is blocked) sticky : true

End RegGroup

1.6.2 KAM_PMR_STATUS

Reg.

0x000C4200 -
0x000C421F

rtl.reg_enb : false
'Count' : 'KAM_PMR_STATUS' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xC4200	0xC4204	0xC4208	...	0xC4214	0xC4218	0xC421C

bits	name	s/w	h/w	default	description
15:0	PMR_VALID	ro	wo	0x0	PMR[n].ENTRY[i] is not valid and will read as zeros VALID[n][i] == 1 : the 32-bit PMR entry, PMR[n].ENTRY[i] valid. It's value will be returned on a read. VALID[n][i] == 0 : the 32-bit PMR entry, PMR[n].ENTRY[i] is not valid. Reading this entry will return 0.

1.6.3 KAM_PMR_SOFT_RESET

Reg. 0x000C4220 - 0x000C423F

rtl.reg_enb : false
'Count' : 'KAM_PMR_SOFT_RESET' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xC4220	0xC4224	0xC4228	...	0xC4234	0xC4238	0xC423C

bits	name	s/w	h/w	default	description
0	PMR_SOFT_RESET	rw	ro	0x0	Set to 1 to invalidate a PMR. None of the PMR_RESET[i] registers may be set after PMR_RESET_LOCK is set self-clearing dontcompare : true 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'PMR_SOFT_RESET' is lock by 'KAM_PMR_SOFT_RESET_LOCK.PMR_SOFT_RESET_LOCK'.

1.6.4 KAM_PMR_SOFT_RESET_LOCK

Reg. 0x000C4240

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
0	PMR_SOFT_RESET_LOCK	rw	na	0x0	Set high to lock the PMR_SOFT_RESET array. This bit is latched high and is on cleared on ROT power-on or warm resets 'lock' : 'PMR_SOFT_RESET_LOCK' is lock by 'KAM_PMR_SOFT_RESET_LOCK.PMR_SOFT_RESET_LOCK'.

1.6.5 KAM_PMR_EXTEND_LOCK

Reg. 0x000C4244 - 0x000C4263

rtl.reg_enb : false
'Count' : 'KAM_PMR_EXTEND_LOCK' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xC4244	0xC4248	0xC424C	...	0xC4258	0xC425C	0xC4260

bits	name	s/w	h/w	default	description
0	PMR_PROTECT	rw	ro	0x0	1: Hardware blocks PMR Extend on PMR[i] - PMR Extend will immediately abort 0: PMR[i] is available for PMR Extend dontcompare : true 'lock' : 'PMR_PROTECT' is lock by 'PMR_PROTECT'. lockRepeat : PMR_PROTECT

1.6.6 KAM_PMR_MEASUREMENT

Reg. 0x000C4264

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	rw	ro	0x0	address of Measurement source, MSMT. HW calulates SHA(PMR | MSMT)

1.6.7 KAM_PMR_EXTEND_CTRL

Reg. 0x000C4268

Writing START = 1 in this register starts the PMR Extend operation.
PMR[PMR_INDEX] = SHA (PMR[PMR_INDEX] || MSG)
rtl.reg_enb : true

bits	name	s/w	h/w	default	description
31	START	r/w1s	rw	0x0	Write 1 to initiate the PMR Extend operation on PMR_INDEX. Harware will clear START when the PMR Extend fuction goes IDLE dontcompare : true rtl.hw_clear : true
30	ABORT	rw	ro	0x0	This control is for unanticipated hang (e.g. PMR Extend function times out) Before setting RESET of this register, firmware should assert ABORT until PMR_EXTEND_STATUS.QUIESCE == 1. Clear the abort before resetting
29	RESET	rw	ro	0x0	Reset the PMR Extend DMA logic. Be sure that PMR_EXTEND_STATUS.QUIESCE == 1 before setting RESET. (see ABORT field) Firmware must set this bit high and then set it low (not self resetting) dontcompare : true
2:0	PMR_INDEX	rw	ro	0x0	Run PMR Extend on PMR[PMR_INDEX]. Be sure to set START to initiate the hardware operation 'lock' : 'PMR_INDEX' is lock by 'START'.

1.6.8 KAM_PMR_EXTEND_STATUS

0x000C426C

rtl.reg_enb : false

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>9</td><td>QUIESCE</td><td>ro</td><td>wo</td><td>0x1</td><td>1: The PMR Extend DMA is quiesced. It is safe to set PMR_EXTEND_CTRL.RESET</td></tr>
<tr><td>8</td><td>BUSY</td><td>ro</td><td>wo</td><td>0x0</td><td>1: A PMR Extend operation is in progress</td></tr>
<tr><td>6:4</td><td>PMR_INDEX</td><td>ro</td><td>wo</td><td>0x0</td><td>PMR Index being extended. Valid only when BUSY == 1</td></tr>
<tr><td>2:0</td><td>PMR_SIZE</td><td>ro</td><td>wo</td><td>0x1</td><td>This is the PMR SIZE read from the SHA Engine
enum:PMR_SIZE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>PMR_256</td><td>1</td><td>PMR is 256-bits (8 PMR Entries)</td></tr>
<tr><td>PMR_384</td><td>2</td><td>PMR is 384-bits (12 PM R Entries)</td></tr>
<tr><td>PMR_512</td><td>4</td><td>PMR is 512-bits (16 PM R Entries)</td></tr>
</table>
encode : PMR_SIZE_e</td></tr>
</table>

1.6.9 KAM_PMR_BUS_ERR_STATUS

0x000C4270

Bus error status for reads - only valid when KAM_ERR_STATUS.PMR_BUS_ERROR == 1
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:2	ADDR	ro	wo	0x0	upper 30 bits of address that caused the bus error
1:0	RESP	ro	wo	0x0	resp from bus

1.6.10 KAM_PMR_SHA_CONFIG

0x000C4274

Writable only by KAM PMR Extend function. Readable by any
rtl.reg_enb : false
no_reg_bit_bash_test : true

display_name : SHA Configuration Mirror
reg_prot : priv
reg_wprot : priv
reg_rprot : priv

bits	name	s/w	h/w	default	description
30:24	MODE	rw	rw	0x0	Mode of operaton of SHA. • 7'b0000001: SHA_256 • 7'b0000010: SHA_384 • 7'b0000100: SHA_512 • 7'b0001000: HMAC_SHA_256 • 7'b0010000: HMAC_SHA_384 • 7'b0100000: HMAC_SHA_512
18:16	KEY_SIZE	rw	rw	0x0	Size of HMAC key. • 3'b001: KEY_128: Use 128-bit key size, input at HMAC_KEY[0-3] • 3'b010: KEY_192: Use 192-bit key size, input at HMAC_KEY[0-5] • 3'b100: KEY_256: Use 256-bit key size, input at HMAC_KEY[0-7]
15:8	IN_LEVEL	rw	ro	0x0	Input FIFO threshold controls SHA_INTR_STATE.IN_FIFO_AE signal. SHA_INTR_STATE.IN_FIFO_AE is latched to 1 when IN_LEVEL is less than or equal to in_fifo_bytes_available
4	DMA_IN	rw	rw	0x0	Enable for DMA request signal for writing input data Use to protect BE Gate signals for power 0: Disable 1: Enable
1	BYTE_SWAP_IN	rw	rw	0x0	Enable endian byte swapping for input data Byte swapping based on this control applies to SHA_MSG, SHA_GEN_DIGEST, SHA_GEN_CONTEXT and SHA_VERIFY, SHA_DIGEST register writes 0: Disable 1: Enable
0	BYTE_SWAP_OUT	rw	rw	0x0	Enable endian byte swapping for output data, applies to SHA_DIGEST register reads only 0: Disable 1: Enable

1.6.11 KAM_PMR_SHA_INTR

Reg. 0x000C4278

Writable only by KAM PMR Extend function. Readable by any
rtl.reg_enb : false
no_reg_bit_bash_test : true
reg_prot : priv
reg_wprot : priv
reg_rprot : priv

bits	name	s/w	h/w	default	description
4	VERIFY_DONE	rw	rw	0x0	Verify done interrupt Status indicatinga hash digest verification following SHA_VERIFY command is complete.
3	GEN_DONE	rw	rw	0x0	Generate done interrupt Status indicating a hash digest or context data is generated following SHA_GEN_DIGEST or SHA_GEN_CONTEXT commands
2	IN_FIFO_AE	rw	rw	0x0	Input FIFO almost empty interrupt The event driving this status can only be changed by software pushing the input FIFO or hardware popping the input FIFO
1	ERR	rw	rw	0x0	Error interrupt Status indicating an error has been detected. This bit is a consolidation(OR)of all error events in SHA_ERR_STATUS.

					The event driving this status can only be changed by software resetting the block.
0	ALERT	rw	rw	0x0	Alert interrupt An security relevant error has been detected. This bit is a consolidation (OR) of all alert events in SHA_ALERT_STATUS. The event driving this status can only be changed by software resetting the block.

1.6.12 KAM_ECC_CTRL_PMR

0x000C427C

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
3	ERR_SB_INSERT	rw	ro	0x0	When set, a PMR write (Extend) followed by read will cause correctable error on bit0 the PMR location
2	ERR_DB_INSERT	rw	ro	0x0	When set, any write to PMR memory using PMR Extend followed by read will cause uncorrectable error on corresponding PMR location.
1	ERR_CLEAR	wo	ro	0x0	When set, will clear any pending ERR_DB or ERR_SB status. This bit is self-clearing 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ENABLE	rw	ro	0x0	1: enables ECC checking on the Instruction RAM memory. The PMR is protected with a 7-bit ECC per 32-bit word. Partial writes can never occur on PMR memory since PMR writes are only done by a hardware state machine.

1.6.13 KAM_ECC_STATUS_PMR

0x000C4280

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31	ERR_DB	ro	wo	0x0	Double bit ECC Error is pending. No other errors will be logged until the field is cleared. Toggle ECC_CTRL_PMR.CLEAR to clear the field and allow future logging of ECC errors
30	ERR_SB	ro	wo	0x0	Single bit ECC Error occured. The field is cleared when ECC_CTRL_PMR.CLEAR is 1
27:24	ERR_COUNT	ro	wo	0x0	ERR_COUNT is incremented each time a single bit ECC error occurs. The count stops at 15. Toggle ECC_CTRL_PMR.CLEAR to clear the count
22:16	ECC	ro	wo	0x0	ECC is the captured syndrome of the single bit error ECC is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware toggles ECC_CTRL_PMR.CLEAR ECC is only valid when ERR_SB and/or ERR_DB are 1
15:12	USER	ro	wo	0xF	USER is the Master Bus USER that triggered the ecc error USER is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware toggles ECC_CTRL_PMR.CLEAR USER is only valid when ERR_SB and/or ERR_DB are 1 USER reflects USER_UNKNOWN when ERR_DB and ERR_SB are 0

enum:ROT_USER_e

Name	Value	Description
USER_SECP_INST	0	SECP Instruction
USER_SECP_DATA	1	SECP Data
USER_SECP_D	2	SECP Debugg

BG		er
USER_EXTP	3	EXTP
USER_KAM	4	KAM PMR Ext end DMA
USER_DMA2	5	ROT DMA 2
USER_DMA3	6	ROT DMA 3
USER_DMA4	7	ROT DMA 4
USER_DMA5	8	ROT DMA 5
USER_NIC_AHBMTX	9	NIC TO AHB_MTX
USER_DMA0	10	DMA 0
USER_DMA1	11	DMA 1
USER_DAP_AXI	13	DAP AXI
USER_UNKNOWN	15	No User

encode : ROT_USER_e

11:0	ADDR	ro	wo	0x0	ADDR is the byte address of ECC error location ADDR is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware toggles ECC_CTRL_PMR.CLEAR ADDR is only valid when ERR_SB and/or ERR_DB are 1

1.6.14 KAM_ECC_LOG_PMR

Reg. 0x000C4284

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	DATA	ro	wo	0x0	Captured Data at ECC error location DATA is captured on last single bit ecc error read access or first double bit ecc error read access. Once a double bit error occurs, logging stops till firmware toggles ECC_CTRL_PMR.CLEAR DATA is only valid when ERR_SB and/or ERR_DB are 1

1.6.15 KAM_OTP_LANE

RegGrp 0x000C4300 - 0x000C437B

'stride' : '16' Specifies the address stride when instantiating an array of components
'Count' : 'KAM_OTP_LANE' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xC4300	0xC4310	0xC4320	...	0xC4350	0xC4360	0xC4370

1.6.15.1 CONFIG

Reg. 0x000C4300

Define a range of OTP rows for the OTP Access Filter. These fields cannot be written when KAM_LANE_ACCESS.OTP_LANE_LOCK is set. OTP Lanes should not overlap (i.e. and OTP Row should reside in 0 or 1 OTP Lanes) - but if an OTP row resides in more than one lane, any READ/PROPGRAM restriction in one lane will be honored
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31	LANE_ENABLE	rw	ro	0x0	The OTP Lane range defined by (LANE_START,LANE_COUNT) is enabled 'lock' : 'LANE_ENABLE' is lock by 'AC-CESS%d.OTP_LANE_LOCK == 1'.
21:16	LANE_COUNT	rw	ro	0x0	LANE_COUNT specifies the number of OTP rows in the LANE Range 'lock' : 'LANE_COUNT' is lock by 'AC-CESS%d.OTP_LANE_LOCK == 1'.
9:0	LANE_START	rw	ro	0x0	LANE_START field specifies OTP Row address of the OTP lane range.

					'lock' : 'LANE_START' is lock by 'ACCESS%d.OTP_LANE_LOCK == 1'.

1.6.15.2 ACCESS

Reg. 0x000C4304

Firmware programs the accesibility of OTP LANE[i] .
Production Mode occurs when

- HW_CONTROL_12 == PRODUCTION, **AND**
- KAM_OAF_EN OTP Hardware Control[10] bit is set

- when 0, RE and WE restrict
  read and write access of an OTP Lane
- all lane controls (ACCESS.RE,ACCESS.WE,CONFIG.*) are locked when OTP_LANE_LOCK high
- In Production Mode, OTP_LANE_LOCK can not be cleared

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
2	OTP_LANE_LOCK	rw	na	0x0	When this bit is set, KAM_OTP_LANE cannot be modified Like RE and WE, this field cannot be cleared when HW_CONTROL_12==PRODUCTION and HW Control[10] KAM_OAF_EN == 1 dontcompare : true 'lock' : 'OTP_LANE_LOCK' is lock by 'ACCESS%d.OTP_LANE_LOCK == 1 && HW_CONTROL_12 == 1'b1 && HW_CONTROL_10 == 1'.
1	RE	rw	ro	0x1	1: Enabled range has read permissions 0: Enable range is not readable (i.e. reads return 0 + read_deny interrupt) 'lock' : 'RE' is lock by 'ACCESS%d.OTP_LANE_LOCK== 1'.
0	WE	rw	ro	0x1	1: Enabled range has write permissions 0: Enable range is not writable (i.e. write ignored + write_deny interrupt) 'lock' : 'WE' is lock by 'ACCESS%d.OTP_LANE_LOCK== 1'.

1.6.15.3 STATUS

Reg. 0x000C4308

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
1	WRITE_DENY	r/w1c	rw	0x0	A program command to OTP matched OTP LANE[i] with LANE_ENABLE == 1 AND WE == 0 rtl.hw_set : true
0	READ_DENY	r/w1c	rw	0x0	A read command to OTP matched an OTP LANE[i] with LANE_ENABLE == 1 AND RE == 0 rtl.hw_set : true

End RegGroup

1.6.16 KAM_OTP_HW_CONTROL_VALID

Reg. 0x000C43C0

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
0	VALID	rw	ro	0x0	When VALID is 1, KAM_OTP_HW_CONTROL values cannot be written. When LCM_MODE == Production and KAM_OTP_HW_CONTROL[9] == 1, VALID is latched high once set, and may only be cleared with a power-on or warm reset

					'lock' : 'VALID' is lock by 'VALID && HW_CONTROL_12 && HW_CONTROL_9'.

1.6.17 KAM_OTP_HW_CONTROL

0x000C43C4

These bits are defined in the ROT Architecure Spec
rtl.reg_enb : false
no_reg_hw_reset_test : true

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>31</td><td>HWC_VALID</td><td>ro</td><td>rw</td><td>0x0</td><td>HWC_VALID is an exact copy of the KAM_OTP_HW_CONTROL_VALID.VALID field
hard_reset : false
resetsignal : kam_por_rstn</td></tr>
<tr><td>30</td><td>AUTOLOAD_VALID</td><td>ro</td><td>rw</td><td>0x0</td><td>HW_CONTROL[4:0] have been autoloaded. This bit is latched high once set
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'AUTOLOAD_VALID' is lock by 'AUTOLOAD_VALID'.</td></tr>
<tr><td>29</td><td>AUTOLOAD_ERR</td><td>ro</td><td>rw</td><td>0x0</td><td>A decode error on HW_CONTRL_1 was detected in the raw rot_otp_autoload[17:3] bits
hard_reset : false
resetsignal : kam_por_rstn</td></tr>
<tr><td>12</td><td>HW_CONTROL_12</td><td>rw</td><td>ro</td><td>0x1</td><td>LCM_MODE
enum:LCM_MODE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>LCM_DEVELOPMENT</td><td>0</td><td>LCM MODE is Development Mode</td></tr>
<tr><td>LCM_PRODUCTION</td><td>1</td><td>LCM MODE is Prodcution Mode</td></tr>
</table>
hard_reset : false
encode : LCM_MODE_e
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_12' is lock by 'HWC_VALID'.</td></tr>
<tr><td>11</td><td>HW_CONTROL_11</td><td>rw</td><td>ro</td><td>0x1</td><td>RoT KAM KV Enable - RoT Chicken bit for enforcing KV range latching
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_11' is lock by 'HWC_VALID'.</td></tr>
<tr><td>10</td><td>HW_CONTROL_10</td><td>rw</td><td>ro</td><td>0x1</td><td>RoT KAM OAF Enable - RoT Chicken bit for enforcing OTP Access Filter Range latching
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_10' is lock by 'HWC_VALID'.</td></tr>
<tr><td>9</td><td>HW_CONTROL_9</td><td>rw</td><td>ro</td><td>0x1</td><td>RoT KAM HW Control Protection enable - RoT Chicken bit for enforcing HW_CONTROL_VALID latching
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_9' is lock by 'HWC_VALID'.</td></tr>
<tr><td>8</td><td>HW_CONTROL_8</td><td>rw</td><td>ro</td><td>0x1</td><td>RSVD
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_8' is lock by 'HWC_VALID'.</td></tr>
<tr><td>7</td><td>HW_CONTROL_7</td><td>rw</td><td>ro</td><td>0x1</td><td>RoT Exclusive Access
1: securty resource access is limited to only the SECP
0: any master has access to all security resources connected to its bus
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_7' is lock by 'HWC_VALID'.</td></tr>
<tr><td>6</td><td>HW_CONTROL_6</td><td>rw</td><td>ro</td><td>0x1</td><td>RSVD
hard_reset : false
resetsignal : kam_por_rstn
'lock' : 'HW_CONTROL_6' is lock by 'HWC_VALID'.</td></tr>
</table>

5	HW_CONTROL_5	rw	rw	0x1	SoC Specific (ESSD only) chicken bit - RoT Soft Reset Disable hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_5' is lock by 'HWC_VALID'.
4	HW_CONTROL_4	rw	rw	0x1	SoC Specific (ESSD only) chicken bit - RoT Report Secure Load Failure Disable hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_4' is lock by 'HWC_VALID'.
3	HW_CONTROL_3	rw	rw	0x1	SoC EXTP Boot Source Selection (ESSD and HDD only) 1: EXTP boot forced from ROM 0: EXTP may boot from ROM or serial flash hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_3' is lock by 'HWC_VALID'.
2	HW_CONTROL_2	rw	rw	0x1	CSSD: SoC UART Disable ESSD: PLL Programming Source Others: RSVD hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_2' is lock by 'HWC_VALID'.
1	HW_CONTROL_1	rw	rw	0x1	SoC Debug Disable 1: disable jtag and other SoC debug features 0: enable jtag and other SoC debug features hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_1' is lock by 'HWC_VALID'.
0	HW_CONTROL_0	rw	rw	0x1	RoT Debug Disable 1: disable debug to RoT 0: enable debug to RoT hard_reset : false resetsignal : kam_por_rstn 'lock' : 'HW_CONTROL_0' is lock by 'HWC_VALID'.

1.6.18 KAM_KV

RegGrp 0x000C4400 - 0x000C4447

'Count' : 'KAM_KV' will repeat '6' times.

count	0	1	2	3	4	5
address	0xC4400	0xC440C	0xC4418	0xC4424	0xC4430	0xC443C

1.6.18.1 RANGE

Reg. 0x000C4400

Define a range in KV_RAM memory for Key Vault. These fields cannot be written when KAM_KV[i].ACCESS.KV_RANGE_LOCK is set
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31	KV_ENABLE	rw	ro	0x0	The range defined by (KV_START,KV_COUNT) is enabled 'lock' : 'KV_ENABLE' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1'.
23:16	KV_COUNT	rw	ro	0x0	KV_COUNT specifies the number of 128-bit KV_RAM entries in the KV Range. Firmware shall ensure that ranges do not overlap and that a range does not exceed the Scratch Memory size 'lock' : 'KV_COUNT' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1'.
11:4	KV_START	rw	ro	0x0	KV_START field specifies the 128-bit start address of the range from the start of KV_RAM memory. The byte address may be written from [11:0] and the bottom 4 bits will be zero regardless of what was written.

					'lock' : 'KV_START' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1'.

1.6.18.2 ACCESS

0x000C4404

Firmware programs the accesibility of KV range[i] (in KV_RAM).
Production Mode occurs when

- HW_CONTROL_12 == PRODUCTION, **AND**
- KAM_KV_EN OTP Hardware Control[11] bit is set

- when 0, RE and WE restrict
  read and write access of an KV Range
- all range controls (ACCESS.RE,ACCESS.WE,RANGE.*) are locked when KV_RANGE_LOCK high
- In Production Mode, KV_RANGE_LOCK can not be cleared

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
2	KV_RANGE_LOCK	rw	na	0x0	When this bit is set, KAM_KV_RANGE cannot be modified Like RE and WE, this field cannot be cleared when HW_CONTROL_12==PRODUCTION and HW Control[11] KAM_KV_EN == 1 dontcompare : true 'lock' : 'KV_RANGE_LOCK' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1 && HW_CONTROL_12 == 1'b1 && HW_CONTROL_11 == 1'.
1	RE	rw	ro	0x1	1: Enabled range has read permissions 0: Enabled range is not readable (i.e. reads return 0 + read_deny interrupt) 'lock' : 'RE' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1'.
0	WE	rw	ro	0x1	1: Enabled range has write permissions 0: Enabled range is not writable (i.e. write ignored + write_deny interrupt) 'lock' : 'WE' is lock by 'ACCESS%d.KV_RANGE_LOCK == 1'.

1.6.18.3 STATUS

0x000C4408

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
1	WRITE_DENY	r/w1c	rw	0x0	A write to KV_RAM matched a KV Range with KV_ENABLE == 1 AND WE == 0 rtl.hw_set : true
0	READ_DENY	r/w1c	rw	0x0	A read from KV_RAM matched a KV Range with KV_ENABLE == 1 AND RE == 0 rtl.hw_set : true

End RegGroup

1.6.19 KAM_ERR_STATUS

0x000C4500

rtl.reg_enb : true

bits	name	s/w	h/w	default	description
5	MPU_READ_VIOLATION	ro	rw	0x0	The PMR Extend Operation has aboorted due to an MPU Read Violation on the Source Message address. To clear status:

					1. Wait for ROT_KAM.KAM_PMR_EXTEND_STATUS.QUIESCE == 1 (expected immediately) 2. Set ROT_KAM.KAM_PMR_EXTEND_CTRL.RESET = 1 3. Set ROT_KAM.KAM_PMR_EXTEND_CTRL.RESET = 0 This status will clear when RESET is set PMR_EXTEND_ABORT is simultaneaously set in this case No PMR values are modified when an PMR_EXTEND_ABORT is set rtl.hw_set : true
4	PMR_BUS_ERROR	ro	rw	0x0	The PMR Extend Operation has aboorted due to a bus error while reading the message. When this field is set, firmware can read bus error location info from PMR_BUS_ERROR register. To clear status: 1. Wait for ROT_KAM.KAM_PMR_EXTEND_STATUS.QUIESCE == 1 (expected immediately) 2. Set ROT_KAM.KAM_PMR_EXTEND_CTRL.RESET = 1 3. Set ROT_KAM.KAM_PMR_EXTEND_CTRL.RESET = 0 This status will clear when RESET is set KAM_ERR_STATUS.PMR_EXTEND_ABORT will always also be set when this field is set. No PMR values are modified when an PMR_EXTEND_ABORT is set rtl.hw_set : true
3	ECC_DB_PMR_ERR	ro	wo	0x0	Double Bit ECC Error. This bit can only be cleared by setting KAM_ECC_CTRL_PMR.ERR_CLEAR. This ECC error can be a result of firmware reading a PMR value directly or it can be set as a result of a PMR_EXTEND operation after hardware has read the PMR into the SHA input FIFO. If the ECC error is active after the PMR_EXTEND hardware has transferred the full PMR value into the SHA engine, the PMR_EXTEND operation will abort. In this case. firmware must reset the SHA Engine, clear the ECC error, and clear the PMR_EXTEND_ABORT status in this case. No values are written to the PMR in this case
2	SHA_ERR_ALERT	r/w1c	rw	0x0	The PMR Extend Operation has aboorted due to a SHA Error or Alert. Firmware should reset the SHA engine and then write 1 to clear this status. PMR_EXTEND_ABORT is simultaneaously set in this case No PMR values are modified when an PMR_EXTEND_ABORT is set rtl.hw_set : true
1	PROTECTED_PMR	r/w1c	rw	0x0	The PMR Extend Operation has aboorted beacuse the PMR[pmr_index] is protected. (i.e. KMA_PMR_EXTEND_LOCK[pmr_index].PMR_PROTECT == 1) PMR_EXTEND_ABORT is simultaneaously set in this case No PMR values are modified when an PMR_EXTEND_ABORT is set rtl.hw_set : true
0	PMR_EXTEND_ABORT	r/w1c	rw	0x0	The PMR Extend operation has aborted. See other fields in this register for the cause rtl.hw_set : true

1.6.20 KAM_INTR_STATE

Reg. 0x000C4504

rtl.reg_enb : true

bits	name	s/w	h/w	default	description
10	OTPC_BUS_ERROR	r/w1c	rw	0x0	A bus error occured while accessing OTP controller. See KAM_OTP_BUS_ERR_STATUS for direction and APB address. Clearing this bit also clears the KAM_OTP_BUS_ERR_STATUS register. rtl.hw_set : true
9	OTPC_INT	r/w1c	rw	0x0	An interrupt from the OTP Controler is set rtl.hw_set : true
8	OTP_WRITE_DENY	r/w1c	rw	0x0	An OTP PROGRAM command write was denied. Refer to KAM_OTP_LANE[lane_index].STATUS.WRITE_DENY rtl.hw_set : true
7	OTP_READ_DENY	r/w1c	rw	0x0	An OTP READ command write was denied. Refer to KAM_OTP_LANE[lane_index].STATUS.READ_DENY rtl.hw_set : true
6	KV_WRITE_DENY	r/w1c	rw	0x0	A KV_RAM memory write was denied due to a KAM_KV Range write restriction. Refer to KAM_KV[kv_index].STATUS.WRITE_DENY rtl.hw_set : true
5	KV_READ_DENY	r/w1c	rw	0x0	A KV_RAM memory read was denied due to a KAM_KV Range read restriction. Refer to KAM_KV[kv_index].STATUS.READ_DENY rtl.hw_set : true
4	SHA_UNAUTHORIZED	r/w1c	rw	0x0	The SHA Engine was accessed by other than the KAM PMR Extend engine while a PMR_EXTEND operation is in progress. The access was ignored by the SHA engine so as not to disturb the PMR Extend operation. rtl.hw_set : true
3	PMR_EXTEND_COMPLETE	r/w1c	rw	0x0	The PMR Extend operation complete without errors rtl.hw_set : true
2	ECC_SB_PMR	r/w1c	rw	0x0	A single bit ECC error is detected from PMR memory read rtl.hw_set : true
1	ERR	r/w1c	rw	0x0	Any of the KAM_ERRO_STATUS bits are set rtl.hw_set : true
0	ALERT	r/w1c	rw	0x0	Can only be set through the INTR_TEST register rtl.hw_set : true

1.6.21 KAM_INTR_ENABLE

Reg. 0x000C4508

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
10	OTPC_BUS_ERROR	rw	ro	0x0	kam_interrupt == INTR_ENABLE.OTPC_BUS_ERROR AND INTR_STATE.OTPC_BUS_ERROR
9	OTPC_INT	rw	ro	0x0	kam interrupt == INTR_ENABLE.OTPC_INT AND INTR_STATE.OTPC_INT
8	OTP_WRITE_DENY	rw	ro	0x0	kam interrupt == INTR_ENABLE.OTP_WRITE_DENY AND INTR_STATE.OTP_WRITE_DENY
7	OTP_READ_DENY	rw	ro	0x0	kam interrupt == INTR_ENABLE.OTP_READ_DENY AND INTR_STATE.OTP_READ_DENY
6	KV_WRITE_DENY	rw	ro	0x0	kam interrupt == INTR_ENABLE.KV_WRITE_DENY AND INTR_STATE.KV_WRITE_DENY
5	KV_READ_DENY	rw	ro	0x0	kam interrupt == INTR_ENABLE.KV_READ_DENY AND INTR_STATE.KV_READ_DENY
4	SHA_UNAUTHORIZED	rw	ro	0x0	kam interrupt == INTR_ENABLE.SHA_UNAUTHORIZED AND INTR_STATE.SHA_UNAUTHORIZED
3	PMR_EXTEND_COMPLETE	rw	ro	0x0	kam interrupt == INTR_ENABLE.PMR_EXTEND_COMPLETE AND INTR_STATE.PMR_EXTEND_COMPLETE
2	ECC_SB_PMR	rw	ro	0x0	kam interrupt == INTR_ENABLE.ECC_SB_PMR AND INTR_STATE.ECC_SB_PMR
1	ERR	rw	ro	0x1	kam interrupt == INTR_ENABLE.ERR AND INTR_STATE.ERR
0	ALERT	rw	ro	0x1	kam interrupt == INTR_ENABLE.ALERT AND INTR_STATE.ALERT

1.6.22 KAM_INTR_TEST

0x000C450C

rtl.reg_enb : false
dontcompare : true

bits	name	s/w	h/w	default	description
10	OTPC_BUS_ERROR	rw	ro	0x0	1: latch INTR_STATE.OTPC_BUS_ERROR high
9	OTPC_INT	rw	ro	0x0	1: latch INTR_STATE.OTPC_INT high
8	OTP_WRITE_DENY	rw	ro	0x0	1: latch INTR_STATE.OTP_WRITE_DENY high
7	OTP_READ_DENY	rw	ro	0x0	1: latch INTR_STATE.OTP_READ_DENY high
6	KV_WRITE_DENY	rw	ro	0x0	1: latch INTR_STATE.KV_WRITE_DENY high
5	KV_READ_DENY	rw	ro	0x0	1: latch INTR_STATE.KV_READ_DENY high
4	SHA_UNAUTHORIZED	rw	ro	0x0	1: latch INTR_STATE.SHA_UNAUTHORIZED high
3	PMR_EXTEND_COMPLETE	rw	ro	0x0	1: latch INTR_STATE.PMR_EXTEND_COMPLETE high
2	ECC_SB_PMR	rw	ro	0x0	1: latch INTR_STATE.ECC_SB_PMR high
1	ERR	rw	ro	0x0	1: latch INTR_STATE.ERR high
0	ALERT	rw	ro	0x0	1: latch INTR_STATE.ALERT high

1.6.23 KAM_OTP_BUS_ERR_STATUS

0x000C4510

Bus error status for reads - only valid when KAM_INTR_STATE.OTPC_BUS_ERROR == 1
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
16	BUS_ERR_VALID	ro	wo	0x0	A Bus Error was detected. This register will not be updated until this bit has been cleared. Clearing INTR_STATE.OTPC_BUS_ERROR clears this register
15	IS_READ	ro	wo	0x0	1: Bus Error occured on an OTP Controller APB read 0: Bus Error occured on an OTP Controller APB write Only valid when BUS_ERR_VALID is set
11:0	ADDR	ro	wo	0x0	12-bits of APB OTP address. Only valid when BUS_ERROR_VALID is set.

1.6.24 KAM_SCRATCH

0x000C4700

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	DATA	rw	na	0x0	general purpose verification register

End RegGroup

1.7 ROT_SHA

0x000D2000 - 0x000D221F

decode_size : 0x00001000
chapter : 1.4, Security Subsystem, SHA Registers
module_name : rot_sha_regs
blockgroup : SECUREPROCESSOR
revision : revision: 02806f1

1.7.1 SHA_CTRL

0x000D2000

General purpose control register
rtl.reg_enb : false
dontcompare : true
display_name : SHA Control

bits	name	s/w	h/w	default	description

2	START	rw	ro	0x0	Control to indicate start of a new operation. When a new configuration of the SHA core is to be used, software should write to this field to indicate core to start processing data after all relevant information has been programmed. For a new hash computation, software should program SHA_CFG, SHA_KEY[0:7] as relevant first, followed by a START along with INIT. For restoring a hash context, software should program SHA_CFG.MODE, SHA_KEY[0:7], SHA_DIGEST[0-15] and SHA_MSG_LEN as relevant first, followed by a START without the INIT. rtl.hw_wp : true resetsignal : SW_Reset 'lock' : 'START' is lock by 'SHA_STATUS.BUSY == 1'b1'.
1	INIT	rw	ro	0x0	Control to initialize the hash. The hash is initialized to start values when this bit is set, and will be cleared by hardware. This bit can only be written when the core is in IDLE state (on reset). START and INIT may be written in a single register access rtl.hw_wp : true resetsignal : SW_Reset 'lock' : 'INIT' is lock by 'SHA_STATUS.BUSY == 1'b1'.
0	SW_RST	rw	ro	0x0	Active high soft reset rtl.hw_wp : true resetsignal : SW_Reset

1.7.2 SHA_CFG

0x000D2004

General purpose configuration register
rtl.reg_enb : false
display_name : SHA Configuration

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>30:24</td><td>MODE</td><td>rw</td><td>ro</td><td>0x1</td><td>Mode of operaton of SHA.

enum:SHA_MODE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>SHA_256</td><td>1</td><td>SHA 256</td></tr>
<tr><td>SHA_384</td><td>2</td><td>SHA 384</td></tr>
<tr><td>SHA_512</td><td>4</td><td>SHA 512</td></tr>
<tr><td>HMAC_SHA_256</td><td>8</td><td>HMAC SHA 256</td></tr>
<tr><td>HMAC_SHA_384</td><td>16</td><td>HMAC SHA 384</td></tr>
<tr><td>HMAC_SHA_512</td><td>32</td><td>HMAC SHA 512</td></tr>
</table>
encode : SHA_MODE_e
dontcompare : true
rtl.hw_wp : false
resetsignal : SW_Reset
'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
'lock' : 'MODE' is lock by 'SHA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>18:16</td><td>KEY_SIZE</td><td>rw</td><td>ro</td><td>0x1</td><td>Size of HMAC key.

enum:HMAC_KEY_SIZE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>KEY_128</td><td>1</td><td>Use 128-bit key size, input at HMAC_KEY[0-3]</td></tr>
<tr><td>KEY_192</td><td>2</td><td>Use 192-bit key size, input at HM</td></tr>
</table></td></tr>
</table>

					AC_KEY[0-5] KEY_256 4 Use 256-bit key size, input at HMAC_KEY[0-7] encode : HMAC_KEY_SIZE_e dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'KEY_SIZE' is lock by 'SHA_STATUS.BUSY == 1'b1'.
15:8	IN_LEVEL	rw	ro	0x0	Input FIFO threshold controls SHA_INTR_STATE.IN_FIFO_AE signal. SHA_INTR_STATE.IN_FIFO_AE is latched to 1 when IN_LEVEL <= in_fifo_bytes_available dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
4	DMA_IN	rw	ro	0x0	Enable for DMA request signal for writing input data Use to protect BE Gate signals for power 0: Disable 1: Enable rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
1	BYTE_SWAP_IN	rw	ro	0x0	Enable endian byte swapping for input data Byte swapping based on this control applies to SHA_MSG, SHA_GEN_DIGEST, SHA_GEN_CONTEXT and SHA_VERIFY, SHA_DIGEST register writes 0: Disable 1: Enable rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	BYTE_SWAP_OUT	rw	ro	0x0	Enable endian byte swapping for output data, applies to SHA_DIGEST register reads only 0: Disable 1: Enable rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.3 SHA_MSG

0x000D2008

Address to write message to input FIFO.

Application constraints:

1. Message must begin on a full-word boundary
2. Data should be written most significant word of the block to least significant word of the block
3. SHA_CFG.BYTE_SWAP_IN set will cause data to be byte-swapped in core

Example: Most likely use case
Intended message: ***0x000102030405060708…10111213***

```
uint32_t message[5] = {
0x00010203,
0x04050607,
0x08090a0b,
0x0c0d0e0f,
0x10111213} ;
```

Input setup and sequence:

```
SHA_GEN_DIGEST_BE    = 0xF;  // all 4 bytes valid in last word
SHA_CFG.BYTE_SWAP_IN = 0;
SHA_MSG.DATA_IN        = message[0]; // 0x00010203
SHA_MSG.DATA_IN        = message[1]; // 0x04050607
SHA_MSG.DATA_IN        = message[2]; // 0x08090a0b
SHA_MSG.DATA_IN        = message[3]; // 0x0c0d0e0f
SHA_GEN_DIGEST.DATA_IN = message[4]; // 0x10111213
```

Example: Byte swapped
Intended message: is *0x000102030405060708...10111213*

```
uint32_t message[5] = {0x03020100,
0x07060504,
0x0b0a0908,
0x0f0e0d0c,
0x13121110} ;
```

Input setup and sequence:

```
SHA_CFG.BYTE_SWAP_IN = 1;
SHA_MSG.DATA_IN        = message[0]; // 0x03020100
SHA_MSG.DATA_IN        = message[1]; // 0x07060504
SHA_MSG.DATA_IN        = message[2]; // 0x0b0a0908
SHA_MSG.DATA_IN        = message[3]; // 0x0f0e0d0c
SHA_GEN_DIGEST.DATA_IN = message[4]; // 0x13121110
```

Example: byte swapped with message length mod 4 == 3
Intended message: *0x000102030405060708090a*

```
uchar_t message[11] = {0x00,0x01,0x02,0x03,
0x04,0x05,0x06,0x07,
0x08,0x09,0x0A      };
uint32_t *p32 = message;
```

Input setup and sequence:

```
SHA_GEN_DIGEST_BE    = 0x7;  // 3 bytes valid in last word
SHA_CFG.BYTE_SWAP_IN = 1;
SHA_MSG.DATA_IN =        p32[0]; // 0x03020100
SHA_MSG.DATA_IN =        p32[1]; // 0x07060504
SHA_GEN_DIGEST.DATA_IN = p32[2]; // 0x--0a0908
```

See SHA_GEN_DIGEST_BE for a list and explanation of all supported formats of the final write to SHA_GEN_DIGEST
rtl.reg_enb : false
'SHA_MSG' is an external.
no_reg_bit_bash_test : true
display_name : SHA Message

bits	name	s/w	h/w	default	description
31:0	DATA_IN	wo	ro	0x0	Input FIFO address for message data Exceptions: Final 1-4 bytes of message is pushed to SHA_GEN_DIGEST Final 4 bytes of a block are pushed to SHA_GEN_CONTEXT when software wants notification that context data is ready for reading. A full word write will cause • push data to the input FIFO • increments the message length by 32-bits rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.4 SHA_GEN_DIGEST

0x000D200C

Address to write last message data to generate hash digest
Application constraints:

1. SHA_CFG.BYTE_SWAP_IN set will cause data to be byte-swapped in core

rtl.reg_enb : false
'SHA_GEN_DIGEST' is an external.
no_reg_bit_bash_test : true

display_name : SHA last Message

bits	name	s/w	h/w	default	description
31:0	DATA_IN	wo	ro	0x0	Input FIFO Address for writing last message data for digest generation Any 32-bit write will cause • push data to the input FIFO • final message length calculation based on SHA_GEN_DIGEST_BE[3:0] • message pad operation • final hash digest generation • drive sha_gen_done interrupt state after hash digest is generated rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.5 SHA_GEN_DIGEST_BE

0x000D2010

<strong>Byte Enables for the final 32-bits of a message</strong>
If not all bytes are valid, they may be left or right justified by hardware depending on the BYTE_SWAP_IN setting.
When data are formatted little endian, hardware always left justifies the data.
For example, consider BYTE_SWAP_IN = 0 with a 6 byte message “abcdef”:
<strong>Little Endian Left justified ending</strong>
<pre>
BYTE_SWAP_IN = 0;
SHA_GEN_DIGEST_BE = 0xC;
SHA_MSG = 0x61626364
SHA_GEN_DIGEST = 0x6566----
</pre>
<strong>Little Endian Right justified ending</strong>
<pre>
BYTE_SWAP_IN = 0;
SHA_GEN_DIGEST_BE = 0x3;
SHA_MSG = 0x61626364
SHA_GEN_DIGEST = 0x----6566 <-- Hardware will left justify
</pre>
When data are formatted big endian, hardware always right justifies the data.
For example, consider BYTE_SWAP_IN = 1 with a 6 byte message “abcdef”:
<strong>Big Endian Right justified ending</strong>
<pre>
BYTE_SWAP_IN = 1;
SHA_GEN_DIGEST_BE = 0x3;
SHA_MSG = 0x64636261
SHA_GEN_DIGEST = 0x----6665
</pre>
<strong>Big Endian Left justified ending</strong>
<pre>
BYTE_SWAP_IN = 1;
SHA_GEN_DIGEST_BE = 0xC;
SHA_MSG = 0x64636261
SHA_GEN_DIGEST = 0x6665---- --> Hardware will right justify
</pre>
rtl.reg_enb : false
display_name : SHA last Message bye enable

bits	name	s/w	h/w	default	description
3:0	GEN_DIGEST_BE	rw	ro	0xF	Byte Enablefor final writeWhen the final bytes of a message are written to SHA_GEN_DIGEST register, this setting tells the IP core how many and which bytes are valid. A 1 on any bit indicates cooresponding byte in SHA_GEN_DIGEST is valid. • Bit 3: SHA_GEN_DIGEST[31:24]

					• Bit 2: SHA_GEN_DIGEST[23:16] • Bit 1: SHA_GEN_DIGEST[15:8] • Bit 0: SHA_GEN_DIGEST[7:0] This would result in counter length increment by 8 for each byte that is valid to compute final message length. Core would add padding in the last block according to number of bytes valid and the message length counter. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.6 SHA_GEN_CONTEXT

Reg. 0x000D2014

Address to write last word of SHA block to generate context.
Application constraints:

1. SHA_CFG.BYTE_SWAP_IN set will cause data to be byte-swapped in core
2. Context generation request may only be written on the last word of a SHA block boundary, which is 64 byte or 128-byte for SHA-256 or SHA-384/512 respectively. In other words, a write to this register should result in message length in multiples of complete SHA block as per the mode, otherwise core will drive a GEN_CONTEXT_ERROR error event.

rtl.reg_enb : false
'SHA_GEN_CONTEXT' is an external.
display_name : SHA last word of block to geberate context

bits	name	s/w	h/w	default	description
31:0	GEN_CONTEXT	wo	ro	0x0	Input FIFO Address for writing last message data for context saving A word write will cause • push data to the input FIFO • increment message bit count by 32-bit • calculate intermediate hash digest • drive sha_gen_done interrupt state after context data is generated OR • drive GEN_CONTEXT_ERROR error if the register is written at the inccorect position in a message (not last word of SHA block boundary) rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.7 SHA_VERIFY

Reg. 0x000D2018

Address to write hash digest for verification. The comparison data may be routed into the input FIFO so that the CPU or a DMA engine need not wait for the digest to be generated before writing the data. Instead, the SHA Core start the comparison by popping the input FIFO once a digest is valid.
SHA_VERIFY may not be written unless SHA_GEN_DIGEST or SHA_GEN_CONTEXT have been written to trigger a hash update.
Application constraints:

1. SHA_VERIFY data is the concatenation of . H[0] is the most significant word of the digest and is always written first
2. SHA_CFG.BYTE_SWAP_IN set will cause data to be byte-swapped in core

rtl.reg_enb : false
'SHA_VERIFY' is an external.
display_name : SHA digest verify data

bits	name	s/w	h/w	default	description
31:0	VERIFY	wo	ro	0x0	Input FIFO Address for writing hash digest for verification Any full word write will cause

					• push hash digest comparison data to the input FIFO • compare hash digest presented to previously generated hash digest using SHA_GEN_DIGEST • drive sha_verify_done interrupt state after digest comparison is complete • set the pass/fail status in SHA_STATUS rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.8 SHA_MSG_LEN

Reg. 0x000D2020 - 0x000D202F

Array to read or write the message length for restoring or saving context.
Application constraints:

1. Counter lengths: SHA-256: 64-bit counter, SHA-384/512: 128-bit counter

rtl.reg_enb : false
display_name : SHA message length for restore/save
'Count' : 'SHA_MSG_LEN' will repeat '4' times.

count	0	1	2	3
address	0xD2020	0xD2024	0xD2028	0xD202C

bits	name	s/w	h/w	default	description
31:0	MSG_LEN	rw	rw	0x0	Message length counter • SHA_256: MSG_LEN [1:0] = Msg length counter bits [63:0] • SHA_384: MSG_LEN [3:0] = Msg length counter bits [127:0] • SHA_512: MSG_LEN [3:0] = Msg length counter bits [127:0] • Contains the computed message length after the SHA_GEN_DIGEST or SHA_GEN_CONTEXT command • Software should write the saved message length to this register to restore computation from that intermediate point onwards. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'MSG_LEN' is lock by 'SHA_STATUS.BUSY == 1'b1'.

1.7.9 SHA_DIGEST

Reg. 0x000D2030 - 0x000D206F

Array of hash digest result data, to be read when generating a digest or to be written to provide context digest to restore
Application constraints:

1. SHA_DIGEST data is the concatenation of n=8 or 16 for SHA-256 and SHA-512 respectively. H[0] is the most significant word of the digest and is always read at the lowest address (i.e. SHA_DIGEST.DIGEST[0])
2. SHA_CFG.BYTE_SWAP_OUT will cause data to be byte-swapped upon read

rtl.reg_enb : false
no_reg_bit_bash_test : true
display_name : SHA Digest
'Count' : 'SHA_DIGEST' will repeat '16' times.

count	1	2	3	...	14	15	16
address	0xD2030	0xD2034	0xD2038	...	0xD2064	0xD2068	0xD206C

bits	name	s/w	h/w	default	description

31:0	DIGEST	rw	rw	0x0	Hash digest result • SHA-256: DIGEST[0:7] • SHA-384: DIGEST[0:11] • SHA-512: DIGEST[0:15] • Contains the computed hash digest result after the SHA_GEN_DIGEST or SHA_GEN_CONTEXT command • Software should write the saved context digest to this register to restore computation from that intermediate point onwards. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'DIGEST' is lock by 'SHA_STATUS.BUSY == 1'b1'.

1.7.10 SHA_STATUS

Reg. 0x000D2070

General purpose status register
rtl.reg_enb : false
display_name : SHA Status register

bits	name	s/w	h/w	default	description
24	VERIFY_FAIL	ro	wo	0x0	Hash verification status Set to 1 if digest verification failed or 0 if digest verification passed following completion of SHA_VERIFY command. This bit will be cleared by hardware after software writes STOP to indicate end of current operation. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
23:16	IN_REQ	ro	wo	0x80	Input FIFO bytes available count IN_REQ is the byte capacity of the FIFO when it's empty rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
8	BUSY	ro	wo	0x0	State of core 0: Idle (state == IDLE) 1: Busy (state !=IDLE) rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	SW_RST_DONE	ro	wo	0x1	Software reset completion status 0: Reset not successful 1: Reset successful rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.11 SHA_ERR_STATUS

Reg. 0x000D2074

General error status register with a status bit for each error event.
For events marked fatal, when detected the block ceases operations and requires a reset before re-use.
For events marked non-fatal, results in special handling within the block as noted and can be cleared by software by writing 1 to it.
rtl.reg_enb : false
display_name : SHA Error Status register

bits	name	s/w	h/w	default	description
16	MSG_LEN_OVERFLOW	r/w1c	rw	0x0	Error state triggered when the message length matches or exceeds the maximum bit count for the SHA mode

					• SHA_256 2^64 • SHA_384 2^128 • SHA_512 2^128 rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
12	VERIFY_SIZE_MISMATCH	r/w1c	rw	0x0	Too many FIFO entries to SHA_VERIFY compared to the SHA DIGEST size causes this error. If too few SHA_VERIFY FIFO entries have been pushed, and the SHA_MSG_IN FIFO is written, this error occurs, the SHA_VERIFY operation results in an error rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
8	GEN_DIGEST_BE_ERR	r/w1c	rw	0x0	Illegal value of 0b0000, 0010, 0100, 0101, 0110, 1001, 1010, 1011, 1101 written rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
4	GEN_CONTEXT_ERR	r/w1c	rw	0x0	Context register error detected(non-fatal) SHA_CONTEXT_LAST register write occurred at the incorrect message position, it must only occur on the last word of a SHA block boundary. Hardware should ignore and discrard the data written at SHA_CONTEXT_LAST in this case, software will need to clear this error status before it can write to SHA_CONTEXT_LAST again. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	IN_FIFO_OVERFLOW	ro	rw	0x0	Input FIFO overflow detected (fatal) rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.12 SHA_ALERT_STATUS

0x000D2078

Security relevant error status register with a status bit for each alertevent.

- For events marked fatal, when detected the block ceases operations and requires a reset before re-use.
- For events marked non-fatal, results in special handling within the block as noted and can be cleared by software by writing 1 to it.

rtl.reg_enb : false
display_name : SHA Alert Status register

bits	name	s/w	h/w	default	description
8	KEY_SIZE_ALERT	ro	wo	0x0	An invalid key size was detected(fatal). If value of SHA_CFG.KEY_SIZE is outside of the legal codedvalues, the core should set thisflag. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	MODE_ALERT	ro	wo	0x0	An invalid mode was detected(fatal). If value of SHA_CFG.MODE is outside of the legal codedvalues, the core should set thisflag. rtl.hw_wp : false resetsignal : SW_Reset

					'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.13 SHA_INTR_STATE

Reg. 0x000D2080

General purpose interrupt status register with each bit corresponding to an interrupt port.
There is 1 bit for consolidated alert and error events each, and a dedicated bit for any other status events that need to be routed as interrupts.
All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be relatched
rtl.reg_enb : false
no_reg_bit_bash_test : true
display_name : SHA Interrupt Status register

bits	name	s/w	h/w	default	description
4	VERIFY_DONE	r/w1c	rw	0x0	Verify done interrupt Status indicating a hash digest verification following SHA_VERIFY command is complete. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
3	GEN_DONE	r/w1c	rw	0x0	Generate done interrupt Status indicating a hash digest or context data is generated following SHA_GEN_DIGEST or SHA_GEN_CONTEXT commands rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
2	IN_FIFO_AE	r/w1c	rw	0x1	Input FIFO almost empty interrupt The event driving this status can only be changed by software pushing the input FIFO or hardware popping the input FIFO rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
1	ERR	r/w1c	rw	0x0	Error interrupt Status indicating an error has been detected. • This bit is a consolidation(OR)of all error events in SHA_ERR_STATUS. • The event driving this status can only be changed by software resetting the block. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ALERT	r/w1c	rw	0x0	Alert interrupt • An security relevant error has been detected. • This bit is a consolidation (OR) of all alert events in SHA_ALERT_STATUS. • The event driving this status can only be changed by software resetting the block. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.14 SHA_INTR_ENABLE

Reg. 0x000D2084

General purpose interrupt enablestatus registerwith a bit to mask/unmask each interrupt port
rtl.reg_enb : false
display_name : SHA Interrupt enable register

bits	name	s/w	h/w	default	description
4	VERIFY_DONE	rw	ro	0x0	Verify done interrupt enable 0: do not generate interrupt 1: generate interrupt if SHA_INT_STATE.VERIFY_DONE rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
3	GEN_DONE	rw	ro	0x0	Generate done interrupt enable 0: do not generate interrupt 1: generate interrupt if SHA_INT_STATE.GEN_DONE rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
2	IN_FIFO_AE	rw	ro	0x0	Input FIFO almost empty interrupt enable 0: do not generate interrupt 1: generate interrupt if SHA_INT_STATE.IN_FIFO_AE dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
1	ERR	rw	ro	0x1	Error interrupt enable 0: do not generate interrupt 1: interrupt if SHA_INT_STATE.ERR rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ALERT	rw	ro	0x1	Alert interrupt enable 0: do not generate interrupt 1: interrupt if SHA_INT_STATE.ALERT rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.15 SHA_INTR_TEST

0x000D2088

General purpose interrupt testregisterwith a bit to force each interrupt for test and debug.
rtl.reg_enb : false
display_name : SHA Interrupt test register

bits	name	s/w	h/w	default	description
4	VERIFY_DONE	rw	ro	0x0	Assert verify done interrupt 0: do not force interrupt 1: force SHA_INTR_STATE.VERIFY_DONE dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
3	GEN_DONE	rw	ro	0x0	Assert generate done interrupt 0: do not force interrupt 1: force SHA_INTR_STATE.GEN_DONE dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
2	IN_FIFO_AE	rw	ro	0x0	Assert input FIFO almost empty interrupt 0: do not force interrupt 1: force SHA_INTR_STATE.IN_FIFO_AE dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset

					'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
1	ERR	rw	ro	0x0	Assert error interrupt 0: do not force interrupt 1: force SHA_INTR_STATE.ERR dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
0	ALERT	rw	ro	0x0	Assert alert interrupt 0: do not force interrupt 1: force SHA_INTR_STATE.ALERT dontcompare : true rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.16 SHA_DBG

Reg. 0x000D2090

SHA Debug Register
rtl.reg_enb : false
'SHA_DBG' is an external.
display_name : SHA Debug Register

bits	name	s/w	h/w	default	description
30:25	IFIFO_FULLNESS_DBG	ro	wo	0x0	Input FIFO fullness level rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
24:17	DMA_IN_REQ_DBG	ro	wo	0x0	DMA in req signal rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
16	DMA_IN_ACK_DBG	ro	wo	0x0	DMA in ack signal rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
15	VERIFY_DONE_DBG	ro	wo	0x0	Verify done interrupt Status indicating a hash digest verification following SHA_VERIFY command is complete. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
14	GEN_DONE_DBG	ro	wo	0x0	Generate done interrupt Status indicating a hash digest or context data is generated following SHA_GEN_DIGEST or SHA_GEN_CONTEXT commands rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
13	IN_FIFO_AE_DBG	ro	wo	0x1	Input FIFO almost empty interrupt The event driving this status can only be changed by software pushing the input FIFO or hardware popping the input FIFO rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
12	ERR_DBG	ro	wo	0x0	Error interrupt Status indicating an error has been detected. • This bit is a consolidation(OR)of all error events in SHA_ERR_STATUS.

					• The event driving this status can only be changed by software resetting the block. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
11	ALERT_DBG	ro	wo	0x0	Alert interrupt • An security relevant error has been detected. • This bit is a consolidation (OR) of all alert events in SHA_ALERT_STATUS. • The event driving this status can only be changed by software resetting the block. rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
10:9	RESET_FSM_DBG	ro	wo	0x0	Reset FSM current state rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
8:7	VERIFY_FSM_DBG	ro	wo	0x0	Verify FSM current state rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
6	RESULT_FSM_DBG	ro	wo	0x0	Result FSM current state rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
5:3	HMAC_FSM_DBG	ro	wo	0x0	HMAC FSM current state rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
2:0	SHA_FSM_DBG	ro	wo	0x0	Main FSM current state rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.

1.7.17 SHA_KEY

0x000D2200 - 0x000D221F

Input key array registers
rtl.reg_enb : false
display_name : SHA HMAC Key
'Count' : 'SHA_KEY' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xD2200	0xD2204	0xD2208	...	0xD2214	0xD2218	0xD221C

bits	name	s/w	h/w	default	description
31:0	HMAC_KEY	rw	ro	0x0	KEY_128: KEY [0-4] = Key bits [127:0] KEY_256: KEY [0-7] = Key bits [255:0] rtl.hw_wp : false resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'HMAC_KEY' is lock by 'SHA_STATUS.BUSY == 1'b1'.

End RegGroup

1.8 ROT_AES

RegGrp 0x000D3000 - 0x000D321F

decode_size : 0x00000800
chapter : 1.3, Security Subsystem, AES Registers
module_name : rot_aes_regs
blockgroup : SECUREPROCESSOR
revision : revision: 02806f1

1.8.1 AES_CTRL

Reg. 0x000D3000

General purpose control register
rtl.reg_enb : false
dontcompare : true
display_name : AES Control

bits	name	s/w	h/w	default	description
2	STOP	wo	ro	0x0	Control to indicate end of current operation. After current operation of AES core is complete, software should write to this field to indicate AES to return to IDLE state. Setting STOP will reset the input and output FIFOs without resetting the registers. resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
1	START	wo	ro	0x0	Control to indicate start of a new operation. When a new configuration of the AES core is to be used, software should write to this field to indicate core to start processing data after all configuration, key and IV information has been input to AES_CFG, AES_KEY[0:7] and AES_IV[3:0] resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'START' is lock by 'AES_STATUS.BUSY == 1'b1'.
0	SW_RST	rw	ro	0x0	Active high soft reset resetsignal : SW_Reset

1.8.2 AES_CFG

Reg. 0x000D3004

General purpose configuration register.
Attribute: All fields only read by HW when
state is IDLE, else ignored
rtl.reg_enb : false
display_name : AES Configuration

bits	name	s/w	h/w	default	description
29:24	IN_LEVEL	rw	ro	0x10	Input FIFO threshold controls AES_INTR_STATE.IN_FIFO_AE signal. AES_INTR_STATE.IN_FIFO_AE is latched to 1 when \s\sIN_LEVEL >= in_fifo_bytes_available dontcompare : true resetsignal : SW_Reset
21:16	OUT_LEVEL	rw	ro	0x10	Output FIFO threshold controls AES_INTR_STATE.OUT_FIFO_AF signal. AES_INTR_STATE.OUT_FIFO_AF is latched to 1 when \s\sOUT_LEVEL >= out_fifo_bytes_available dontcompare : true resetsignal : SW_Reset
15:8	MODE	rw	ro	0x0	Mode of operation (Use hamming encoded for following valid configurations) enum:AES_COMMAND_e

Name	Value	Description

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td><table><tr><td>ECB_ENC</td><td>0</td><td>ECB Encrypt</td></tr><tr><td>ECB_DEC</td><td>15</td><td>ECB Decrypt</td></tr><tr><td>CBC_ENC</td><td>22</td><td>CBC Encrypt</td></tr><tr><td>CBC_DEC</td><td>25</td><td>CBC Decrypt</td></tr><tr><td>CTR</td><td>42</td><td>CTR</td></tr><tr><td>GCM_ENC</td><td>51</td><td>GCM Encrypt</td></tr><tr><td>GCM_DEC</td><td>60</td><td>GCM Decrypt</td></tr><tr><td>CCM_ENC</td><td>67</td><td>CCM Encrypt</td></tr><tr><td>CCM_DEC</td><td>76</td><td>CCM Decrypt</td></tr><tr><td>RESERVED</td><td>128</td><td>Reserved</td></tr></table>encode : AES_COMMAND_e
dontcompare : true
resetsignal : SW_Reset
'lock' : 'MODE' is lock by 'AES_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>6:4</td><td>KEY_SIZE</td><td>rw</td><td>ro</td><td>0x4</td><td>Size of key
(Use hamming encoded or one-hot values for following valid configurations)

enum:AES_KEY_SIZE_e<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>KEY_128</td><td>1</td><td>128-bit Key AES_KEY[0-4]</td></tr><tr><td>KEY_192</td><td>2</td><td>Not supported reserved for future use</td></tr><tr><td>KEY_256</td><td>4</td><td>256-bit Key AES_KEY[0-7]</td></tr></table>encode : AES_KEY_SIZE_e
dontcompare : true
rtl.hw_wp : true
resetsignal : SW_Reset
'lock' : 'KEY_SIZE' is lock by 'AES_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>3</td><td>DMA_IN</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable for DMA request signal for writing input data
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
<tr><td>2</td><td>DMA_OUT</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable for DMA request signal for reading output data
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
<tr><td>1</td><td>BYTE_SWAP_IN</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable endian byte swapping for input data
Byte swapping based on this control applies to AES_DATA_IN register writes
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
<tr><td>0</td><td>BYTE_SWAP_OUT</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable endian byte swapping for output data
Byte swapping based on this control applies to AES_DATA_OUT register writes
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
</table>

1.8.3 AES_P_LEN_0

0x000D3008

Used for GCM/CCM mode for len(P/C) - DW MSB, where Programmable from 0 to 2^39-256 bits
rtl.reg_enb : false
display_name : AES Payload Length

bits	name	s/w	h/w	default	description
6:0	P_LEN_0	rw	ro	0x0	Payload Length [38:32], located in PAYLOAD_LEN_0[6:0] resetsignal : SW_Reset 'lock' : 'P_LEN_0' is lock by 'AES_STATUS.BUSY == 1'b1'.

1.8.4 AES_P_LEN_1

Reg. 0x000D300C

Used for GCM/CCM mode for len(P/C) - DW MSB, where Programmable from 0 to 2^39-256 bits
rtl.reg_enb : false
display_name : AES Payload Length

bits	name	s/w	h/w	default	description
31:0	P_LEN_1	rw	ro	0x0	Payload Length [31: 0] resetsignal : SW_Reset 'lock' : 'P_LEN_1' is lock by 'AES_STATUS.BUSY == 1'b1'.

1.8.5 AES_AAD_LEN

Reg. 0x000D3010 - 0x000D3017

Used for GCM/CCM mode for len(AAD). Programmable from 0 to 2^64-1 bits
rtl.reg_enb : false
display_name : AES AAD Length
'Count' : 'AES_AAD_LEN' will repeat '2' times.

count	0	1
address	0xD3010	0xD3014

bits	name	s/w	h/w	default	description
31:0	AAD_LEN	rw	ro	0x0	AAD_LEN[i] AAD_LEN[0]: AAD Length [63:32] AAD_LEN[1]: AAD Length [31: 0] resetsignal : SW_Reset 'lock' : 'AAD_LEN' is lock by 'AES_STATUS.BUSY == 1'b1'.

1.8.6 AES_IV

Reg. 0x000D3018 - 0x000D3027

Initialization vector array
rtl.reg_enb : false
display_name : AES Initialization Vector
'Count' : 'AES_IV' will repeat '4' times.

count	0	1	2	3
address	0xD3018	0xD301C	0xD3020	0xD3024

bits	name	s/w	h/w	default	description
31:0	IV	rw	ro	0x0	IV[i] IV[0]: Initialization Vector [127:96] IV[1]: Initialization Vector [95:64] IV[2]: Initialization Vector [63:32] IV[3]: Initialization Vector [31:0] resetsignal : SW_Reset 'lock' : 'IV' is lock by 'AES_STATUS.BUSY == 1'b1'.

1.8.7 AES_DATA_IN

Reg. 0x000D3028

Address to write to input data FIFO.
Application constraints:
1. Data should be written most significant word of the block to least significant word of the block
2. AES_CFG.BYTE_SWAP_IN set will cause data to be byte-swapped in core
Example: For a 128-bit input block: 0x00112233445566778899aabbccddeeff
Input sequence for 32-bit data bus:
\s\s\s\s 0x00112233
\s\s\s\s 0x44556677
\s\s\s\s 0x8899aabb
\s\s\s\s 0xccddeeff
rtl.reg_enb : false
'AES_DATA_IN' is an external.
no_reg_bit_bash_test : true
display_name : AES DATA Input FIFO

bits	name	s/w	h/w	default	description
31:0	DATA_IN	wo	ro	0x0	Input FIFO address Data written to this address is pushed into the Input FIFO. In AES GCM mode, first all AAC data should be written, and only then - the plaintext. In AES CCM mode, the order of witten data is the following: B0, formatted AAC data and only then - the plaintext resetsignal : SW_Reset

1.8.8 AES_DATA_OUT

0x000D302C

Read top of Output FIFO - Do Not POP on reads
rtl.reg_enb : false
'AES_DATA_OUT' is an external.
no_reg_bit_bash_test : true
display_name : AES DATA Output FIFO

bits	name	s/w	h/w	default	description
31:0	DATA_OUT	rw	wo	0x0	Output FIFO address Reads return the top of the FIFO but do not pop the FIFO In AES GCM/CCM mode, last read returns the AES-GCM autentication TAG Write any value to pop the FIFO sticky : true resetsignal : SW_Reset

1.8.9 AES_STATUS

0x000D3040

General purpose status register
rtl.reg_enb : false
display_name : AES Status

bits	name	s/w	h/w	default	description
29:24	OUT_REQ	ro	wo	0x0	Output FIFO bytes available count OUT_REQ is the byte capacity of the FIFO when it's full resetsignal : SW_Reset
21:16	IN_REQ	ro	wo	0x20	Input FIFO bytes available count IN_REQ is the byte capacity of the FIFO when it's empty resetsignal : SW_Reset
1	BUSY	ro	wo	0x0	State of core 0: Idle (state == IDLE) 1: Busy (state != IDLE) resetsignal : SW_Reset
0	SW_RST_DONE	ro	wo	0x1	Software reset completion status 0: Reset not successful 1: Reset successful resetsignal : SW_Reset

1.8.10 AES_ERR_STATUS

0x000D3044

General purpose status register
rtl.reg_enb : false
display_name : AES Error Status

bits	name	s/w	h/w	default	description
1	OUT_FIFO_UNDERFLOW	ro	rw	0x0	Output FIFO underflow detected must reset to clear resetsignal : SW_Reset
0	IN_FIFO_OVERFLOW	ro	rw	0x0	Input FIFO overflow detected must reset to clear resetsignal : SW_Reset

1.8.11 AES_ALERT_STATUS

0x000D3048

Security relevant error status register with a status bit for each alert event.

rtl.reg_enb : false
display_name : AES Alert Status

bits	name	s/w	h/w	default	description
1	KEY_SIZE_ALERT	ro	wo	0x0	An invalid key size was detected. If value of AES_CFG.KEY_SIZE is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset
0	MODE_ALERT	ro	wo	0x0	An invalid mode was detected. If value of AES_CFG.MODE is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset

1.8.12 AES_INTR_STATE

Reg. 0x000D3050

General purpose interrupt status register with each bit corresponding to an interrupt port. There is 1 bit for consolidated alert and error events each, and a dedicated bit for any other status events that need to be routed as interrupts. All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be re-latched.
rtl.reg_enb : false
no_reg_bit_bash_test : true
display_name : AES Interrupt State

bits	name	s/w	h/w	default	description
3	OUT_FIFO_AF	r/w1c	rw	0x0	Output FIFO almost full interrupt The event driving this status can only be changed by software popping the output FIFO or hardware pushing the output FIFO. rtl.hw_set : true resetsignal : SW_Reset
2	IN_FIFO_AE	r/w1c	rw	0x1	Input FIFO almost empty interrupt The event driving this status can only be changed by software pushing the input FIFO or hardware popping the input FIFO. rtl.hw_set : true resetsignal : SW_Reset
1	ERR	r/w1c	rw	0x0	Status indicating an error has been detected. This bit is a consolidation (OR) of all error events in AES_ERR_STATUS. The event driving this status can only be changed by software resetting the block. rtl.hw_set : true resetsignal : SW_Reset
0	ALERT	r/w1c	rw	0x0	A security relevant error has been detected. This bit is a consolidation (OR) of all alert events in AES_ALERT_STATUS. The event driving this status can only be changed by software resetting the block. rtl.hw_set : true resetsignal : SW_Reset

1.8.13 AES_INTR_ENABLE

Reg. 0x000D3054

General purpose interrupt enable status register with a bit to mask/unmask each interrupt port.
rtl.reg_enb : false
display_name : AES Interrupt Enable
'intr.mask' : Identifies the 'AES_INTR_STATE' for the interrupt logic.

bits	name	s/w	h/w	default	description
3	OUT_FIFO_AF	rw	ro	0x0	Output FIFO almost full interrupt enable 0: do not generate interrupt 1: generate interrupt if AES_INT_STATE.OUT_FIFO_AF resetsignal : SW_Reset

2	IN_FIFO_AE	rw	ro	0x0	Input FIFO almost empty interrupt enable 0: do not generate interrupt 1: generate interrupt if AES_INT_STATE.IN_FIFO_AE dontcompare : true resetsignal : SW_Reset
1	ERR	rw	ro	0x1	Error interrupt enable 0: do not generate interrupt 1: generate interrupt if AES_INT_STATE.ERR resetsignal : SW_Reset
0	ALERT	rw	ro	0x1	Alert interrupt enable 0: do not generate interrupt 1: generate interrupt if AES_INT_STATE.ALERT resetsignal : SW_Reset

1.8.14 AES_INTR_TEST

Reg. 0x000D3058

General purpose interrupt test register with a bit to force each interrupt for test and debug.
rtl.reg_enb : false
display_name : AES Interrupt Test

bits	name	s/w	h/w	default	description
3	OUT_FIFO_AF	rw	ro	0x0	Assert output FIFO almost full interrupt 0: do not force interrupt 1: force AES_INTR_STATE.OUT_FIFO_AF dontcompare : true resetsignal : SW_Reset
2	IN_FIFO_AE	rw	ro	0x0	Assert input FIFO almost empty interrupt 0: do not force interrupt 1: force AES_INTR_STATE.IN_FIFO_AE dontcompare : true resetsignal : SW_Reset
1	ERR	rw	ro	0x0	Assert error interrupt 0: do not force interrupt 1: force AES_INTR_STATE.ERR dontcompare : true resetsignal : SW_Reset
0	ALERT	rw	ro	0x0	Assert alert interrupt 0: do not force interrupt 1: force AES_INTR_STATE.ALERT dontcompare : true resetsignal : SW_Reset

1.8.15 AES_DEBUG

Reg. 0x000D3064

AES Debug register. Hold valid values when i_aes_debug_en==1. Otherwise return 0.
rtl.reg_enb : false
display_name : AES DEBUG regsiter (valid when i_aes_debug_en==1)

bits	name	s/w	h/w	default	description
18	DMA_OUT_ACK	ro	wo	0x0	Output FIFO is being popped Valid either when AES_CFG.DMA_OUT is 0 or 1 resetsignal : SW_Reset
17	DMA_IN_ACK	ro	wo	0x0	Input FIFO is being pushed Valid either when AES_CFG.DMA_IN is 0 or 1 resetsignal : SW_Reset
16:11	DMA_OUT_REQ	ro	wo	0x0	Output FIFO bytes available count Valid either when AES_CFG.DMA_OUT is 0 or 1 resetsignal : SW_Reset
10:5	DMA_IN_REQ	ro	wo	0x20	Input FIFO bytes available count Valid either when AES_CFG.DMA_IN is 0 or 1 resetsignal : SW_Reset
4	OUT_FIFO_AF	ro	wo	0x0	Output FIFO almost full interrupt Valid either when AES_INTR_ENABLE.OUT_FIFO_AF is 0 or 1 resetsignal : SW_Reset
3	IN_FIFO_AE	ro	wo	0x1	Input FIFO almost empty interrupt

					Valid either when AES_INTR_ENABLE.IN_FIFO_AE is 0 or 1 resetsignal : SW_Reset
2:0	AES_STATE	ro	wo	0x0	The state of the ROT AES. Valid only when i_aes_debug_en==1. 000: IDLE 001: KEY_BUSY 010: AES_ACTIVE 011: ABORT 100: GCM_H_BUSY resetsignal : SW_Reset

1.8.16 AES_DATA_OUT_POP

Reg. 0x000D311C

Output FIFO address
Reads return the top of the FIFO and pops the FIFO
In AES GCM mode, last read returns the AES-GCM autentication TAG
rtl.reg_enb : false
'AES_DATA_OUT_POP' is an external.
no_reg_bit_bash_test : true
display_name : AES DATA Output FIFO

bits	name	s/w	h/w	default	description
31:0	DATA_OUT	ro	wo	0x0	Output FIFO address Reads return the top of the FIFO and pops the FIFO In AES GCM/CCM mode, last read returns the AES-GCM/CCM autentication TAG Write any value to pop the FIFO resetsignal : SW_Reset

1.8.17 AES_KEY

Reg. 0x000D3200 - 0x000D321F

Initialization vector array
rtl.reg_enb : false
display_name : AES Initialization Vector
'Count' : 'AES_KEY' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xD3200	0xD3204	0xD3208	...	0xD3214	0xD3218	0xD321C

bits	name	s/w	h/w	default	description
31:0	KEY	rw	ro	0x0	key for encryption/decryption rtl.hw_wp : true resetsignal : SW_Reset 'lock' : 'KEY' is lock by 'AES_STATUS.BUSY == 1'b1'.

End RegGroup

1.9 GCE_REGS_SYS

RegGrp 0x000D3800 - 0x000D38F3

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000800
chapter : 1.36, Security Subsystem, CS AES GCM
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.9.1 gce_regs

RegGrp 0x000D3800 - 0x000D38F3

1.9.1.1 GCE_AES

Reg. 0x000D3800

bits	name	s/w	h/w	default	description
0	SW_RESET	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.2 GCE_CTRL

Reg. 0x000D3808

bits	name	s/w	h/w	default	description
0	START	wo	rw	0x0	
1	STOP	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.3 GCE_CFG

Reg. 0x000D3810

bits	name	s/w	h/w	default	description
0	BYTE_SWAP_OUT	rw	rw	0x0	
1	BYTE_SWAP_IN	rw	rw	0x0	
2	WORD_SWAP_OUT	rw	rw	0x0	
3	WORD_SWAP_IN	rw	rw	0x0	
4	DMA_OUT	rw	rw	0x0	
5	DMA_IN	rw	rw	0x0	
6	KEY_SIZE	rw	rw	0x0	
8:7	MODE	rw	rw	0x0	
9	ENC_DEC	rw	rw	0x0	
15:10	OUT_LEVEL	rw	rw	0x10	
21:16	IN_LEVEL	rw	rw	0x10	
31	reserved_31	ro	rw	0x0	

1.9.1.4 GCE_IV0

Reg. 0x000D3818

bits	name	s/w	h/w	default	description
31:0	IV	rw	rw	0x0	

1.9.1.5 GCE_IV1

Reg. 0x000D3820

bits	name	s/w	h/w	default	description
31:0	IV	rw	rw	0x0	

1.9.1.6 GCE_IV2

Reg. 0x000D3828

bits	name	s/w	h/w	default	description
31:0	IV	rw	rw	0x0	

1.9.1.7 GCE_IV3

Reg. 0x000D3830

bits	name	s/w	h/w	default	description
31:0	IV	rw	rw	0x0	

1.9.1.8 GCE_KEY0

Reg. 0x000D3838

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.9 GCE_KEY1

Reg. 0x000D3840

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.10 GCE_KEY2

Reg. 0x000D3848

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.11 GCE_KEY3

Reg. 0x000D3850

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.12 GCE_KEY4

Reg. 0x000D3858

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.13 GCE_KEY5

Reg. 0x000D3860

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.14 GCE_KEY6

Reg. 0x000D3868

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.15 GCE_KEY7

Reg. 0x000D3870

bits	name	s/w	h/w	default	description
31:0	KEY	rw	rw	0x0	

1.9.1.16 GCE_DATA_IN

Reg. 0x000D3878

bits	name	s/w	h/w	default	description
31:0	DATA_IN	wo	rw	0x0	

1.9.1.17 GCE_DATA_OUT

Reg. 0x000D3880

bits	name	s/w	h/w	default	description
31:0	DATA_OUT	rw	rw	0x0	

1.9.1.18 GCE_STATUS

Reg. 0x000D3888

bits	name	s/w	h/w	default	description

0	SW_RST_DONE	ro	rw	0x1	
1	BUSY	ro	rw	0x0	
7:2	IN_REQ	ro	rw	0x20	
13:8	OUT_REQ	ro	rw	0x0	
14	IN_ACK	ro	rw	0x0	
15	OUT_ACK	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.19 GCE_ERR_STATUS

Reg. 0x000D3890

bits	name	s/w	h/w	default	description
0	OUT_FIFO_UNDERFLOW	ro	rw	0x0	
1	IN_FIFO_OVERFLOW	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.20 GCE_ALERT_STATUS

Reg. 0x000D3898

bits	name	s/w	h/w	default	description
0	MODE_ALERT	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.21 GCE_INTR_STATE

Reg. 0x000D38A0

bits	name	s/w	h/w	default	description
0	ALERT	r/w1c	rw	0x0	
1	ERR	r/w1c	rw	0x0	
2	IN_FIFO_AE	r/w1c	rw	0x1	
3	OUT_FIFO_AF	r/w1c	rw	0x0	
4	DONE	r/w1c	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.22 GCE_INTR_ENABLE

Reg. 0x000D38A8

bits	name	s/w	h/w	default	description
0	ALERT	rw	rw	0x1	
1	ERR	rw	rw	0x1	
2	IN_FIFO_AE	rw	rw	0x0	
3	OUT_FIFO_AF	rw	rw	0x0	
4	DONE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.23 GCE_DATA_OUT_POP

Reg. 0x000D38B0

bits	name	s/w	h/w	default	description
31:0	DATA_OUT	ro	rw	0x0	

1.9.1.24 GCE_ADD_LENGTH

Reg. 0x000D38B8

bits	name	s/w	h/w	default	description
15:0	LENA	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.25 GCE_LAST

Reg. 0x000D38C0

bits	name	s/w	h/w	default	description
0	LAST	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.26 GCE_BCN

Reg. 0x000D38C8

bits	name	s/w	h/w	default	description
2:0	BCN	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.27 GCE_QBCN

Reg. 0x000D38D0

bits	name	s/w	h/w	default	description
2:0	QBCN	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.9.1.28 GCE_TAG0

Reg. 0x000D38D8

bits	name	s/w	h/w	default	description
31:0	TAG	ro	rw	0x0	

1.9.1.29 GCE_TAG1

Reg. 0x000D38E0

bits	name	s/w	h/w	default	description
31:0	TAG	ro	rw	0x0	

1.9.1.30 GCE_TAG2

Reg. 0x000D38E8

bits	name	s/w	h/w	default	description
31:0	TAG	ro	rw	0x0	

1.9.1.31 GCE_TAG3

Reg. 0x000D38F0

bits	name	s/w	h/w	default	description
31:0	TAG	ro	rw	0x0	

End RegGroup

End RegGroup

1.10 ROT_ECA

RegGrp 0x000D4000 - 0x000D4E77

decode_size : 0x00001000
chapter : 1.5, Security Subsystem, Eliptic Curve Accelerator (ECA) Registers
module_name : rot_eca_regs
blockgroup : SECUREPROCESSOR
revision : revision: 02806f1

1.10.1 ECA_MEM

Reg. 0x000D4000 - 0x000D4DFF

Segment access.

rtl.reg_enb : false

'ECA_MEM' is an external.

no_reg_tests : true

display_name : ECA Memory Read Write

'Count' : 'ECA_MEM' will repeat '896' times.

count	1	2	3	...	894	895	896
address	0xD4000	0xD4004	0xD4008	...	0xD4DF4	0xD4DF8	0xD4DFC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.10.2 ECA_CTRL

Reg. 0x000D4E00

General purpose control register

rtl.reg_enb : false

display_name : ECA Control

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>31:24</td><td>RESULT_INDEX</td><td>wo</td><td>ro</td><td>0x0</td><td>Index at which output result of operation must be written in the memory.
Result index may be same asaninput operand index, if software wants to overwrite a location.
After operation is complete, hardware writes the result at this index and
sets the ECA_STATUS.DONE flag.
resetsignal : SW_Reset
'lock' : 'RESULT_INDEX' is lock by 'ECA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>23:16</td><td>OPD2_INDEX</td><td>wo</td><td>ro</td><td>0x0</td><td>Index of the second operand (Y) in memory.
Indicates memory lane index at which second operand is located.
resetsignal : SW_Reset
'lock' : 'OPD2_INDEX' is lock by 'ECA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>15:8</td><td>OPD1_INDEX</td><td>wo</td><td>ro</td><td>0x0</td><td>Index of the first operand (X) in memory.
Indicates memory lane index at which first operand is located.
resetsignal : SW_Reset
'lock' : 'OPD1_INDEX' is lock by 'ECA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>7:4</td><td>CMD</td><td>wo</td><td>ro</td><td>0x0</td><td>Control to indicate command for operation
(Use hamming encoded or one-hot values for
following valid configurations)

enum:ECA_COMMAND_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>ADD</td><td>1</td><td>Addition modulo-p:
x+y mod(p)</td></tr>
<tr><td>ADD_INV</td><td>2</td><td>Additive inverse modulo-p:
-x mod(p)</td></tr>
<tr><td>MULT</td><td>4</td><td>Multiplication modulo-p:
xy mod(p)</td></tr>
<tr><td>MULT_INV</td><td>8</td><td>Multiplicative inverse modulo-p:
x^-1 mod(p)</td></tr>
</table>
encode : ECA_COMMAND_e
resetsignal : SW_Reset
'lock' : 'CMD' is lock by 'ECA_STATUS.BUSY == 1'b1'.</td></tr>
</table>

1	START	wo	ro	0x0	Control to indicate start of a new operation. Software should write to this field to indicate core to start operation indicated in ECA_CFG.CMD resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'START' is lock by 'ECA_STATUS.BUSY == 1'b1'.
0	SW_RST	wo	ro	0x0	Active high soft reset rtl.hw_wp : true

1.10.3 ECA_CFG

Reg. 0x000D4E04

General purpose configuration register.
Bit length of operands, legal values are 128 to 512 bits
rtl.reg_enb : false
display_name : ECA Configuration

bits	name	s/w	h/w	default	description
13:4	OPD_SIZE	rw	ro	0x0	Size of the operand to use . rtl.hw_wp : true resetsignal : SW_Reset 'lock' : 'OPD_SIZE' is lock by 'ECA_STATUS.BUSY == 1'b1'.
1	BYTE_SWAP_IN	rw	ro	0x0	Enable endian byte swapping for input data Byte swapping based on this control applies to ECA memory segments writes 0: Disable 1: Enable resetsignal : SW_Reset
0	BYTE_SWAP_OUT	rw	ro	0x0	Enable endian byte swapping for output data Byte swapping based on this control applies to ECA memory segments reads 0: Disable 1: Enable resetsignal : SW_Reset

1.10.4 ECA_P_UNCOMP

Reg. 0x000D4E08 - 0x000D4E47

Registers array for the uncompressed 'p' format
rtl.reg_enb : false
display_name : Uncompressed 'p'
'Count' : 'ECA_P_UNCOMP' will repeat '16' times.

count	1	2	3	...	14	15	16
address	0xD4E08	0xD4E0C	0xD4E10	...	0xD4E3C	0xD4E40	0xD4E44

bits	name	s/w	h/w	default	description
31:0	P_OPD	rw	ro	0x0	Uncompressed 'P' operand: P_OPD [0]: Polynomial [31:0] P_OPD [1]: Polynomial [63:32] P_OPD [2]: Polynomial [95:64] ... P_OPD [15]: Polynomial [511:479] 'lock' : 'P_OPD' is lock by 'ECA_STATUS.BUSY == 1'b1'.

1.10.5 ECA_P_COEF_0

Reg. 0x000D4E48

Register for compressed 'p' format - 32 LSBs of the Polynomial representation
rtl.reg_enb : false
display_name : Compressed 'p' - reg 0

bits	name	s/w	h/w	default	description
31:0	P_COEF_0	rw	ro	0x0	32 LSBs of the polynomial 'lock' : 'P_COEF_0' is lock by 'ECA_STATUS.BUSY == 1'b1'.

1.10.6 ECA_P_COEF_1

Reg. 0x000D4E4C

Register for compressed 'p' format - 10 MSBs of the Polynomial representation
rtl.reg_enb : false
display_name : Compressed 'p' - reg 1

bits	name	s/w	h/w	default	description
9:0	P_COEF_1	rw	ro	0x0	10 MSBs of the polynomial 'lock' : 'P_COEF_1' is lock by 'ECA_STATUS.BUSY == 1'b1'.

1.10.7 ECA_P_PARAMS

Reg. 0x000D4E50

Register for parameters related to the compressed format of 'p'
rtl.reg_enb : false
display_name : Compressed 'p' parameters

bits	name	s/w	h/w	default	description
22:12	DIGIT_LEN	rw	ro	0x0	Length of a digit. Supports any value between 32 and 521 'lock' : 'DIGIT_LEN' is lock by 'ECA_STATUS.BUSY == 1'b1'.
8:4	HIGHEST_DEG	rw	ro	0x0	Highest degree of the polynomial. Support any value between 1 and 22 'lock' : 'HIGHEST_DEG' is lock by 'ECA_STATUS.BUSY == 1'b1'.
0	FLAG	rw	ro	0x0	Determines the set of coefficients of the poly representation in P_COEFF_1&2: 0: coeff = {-1,0,1} 1: coeff = -31 to 32 'lock' : 'FLAG' is lock by 'ECA_STATUS.BUSY == 1'b1'.

1.10.8 ECA_STATUS

Reg. 0x000D4E54

General purpose status register
rtl.reg_enb : false
display_name : ECA Status

bits	name	s/w	h/w	default	description
1	BUSY	ro	rw	0x0	State of core 0: Idle (state == IDLE) 1: Busy (state != IDLE) resetsignal : SW_Reset
0	SW_RST_DONE	ro	wo	0x1	Software reset completion status 0: Reset not successful 1: Reset successful rtl.hw_set : true rtl.hw_clear : true

1.10.9 ECA_ERR_STATUS

Reg. 0x000D4E58

General error status register with status bit for each error event
rtl.reg_enb : false
display_name : ECA Error Status

bits	name	s/w	h/w	default	description
9	ECA_SERR	r/w1c	rw	0x0	Single bit error detected and corrected by SecDed (non-fatal) resetsignal : SW_Reset
8	ECA_DERR	ro	rw	0x0	Double bit error detected by SecDed (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. resetsignal : SW_Reset

4	Y_ERR	ro	wo	0x0	Invalid Y value detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * Y_ERR: Y >= P resetsignal : SW_Reset
1:0	X_ERR	ro	wo	0x0	Invalid X value detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * X_ERR[0]: X >= P * X_ERR[1]: X == 0 resetsignal : SW_Reset

1.10.10 ECA_ALERT_STATUS

Reg. 0x000D4E5C

Security relevant error status register with a status bit for each alert event.
rtl.reg_enb : false
display_name : ECA Alert Status

bits	name	s/w	h/w	default	description
0	CMD_ALERT	ro	wo	0x0	An invalid modulus size was detected (fatal). If value of ECA_CFG.CMD is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset

1.10.11 ECA_INTR_STATE

Reg. 0x000D4E60

General purpose interrupt status register with each bit corresponding to an interrupt port. There is 1 bit for consolidated alert and error events each, and a dedicated bit for any other status events that need to be routed as interrupts. All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be re-latched.
rtl.reg_enb : false
display_name : ECA Interrupt State

bits	name	s/w	h/w	default	description
2	DONE	r/w1c	rw	0x0	Command done interrupt. Status indicating current operation is done, and results are ready to be read. rtl.hw_set : true resetsignal : SW_Reset
1	ERR	r/w1c	rw	0x0	Error interrupt. Status indicating an error has been detected. This bit is a consolidation (OR) of all error events in ECA_ERR_STATUS. rtl.hw_set : true resetsignal : SW_Reset
0	ALERT	r/w1c	rw	0x0	Alert interrupt. A security relevant error has been detected. This bit is a consolidation (OR) of all alert events in ECA_ALERT_STATUS.. rtl.hw_set : true resetsignal : SW_Reset

1.10.12 ECA_INTR_ENABLE

Reg. 0x000D4E64

General purpose interrupt enable status register with a bit to mask/unmask each interrupt port.
rtl.reg_enb : false
display_name : ECA Interrupt Enable
'intr.mask' : Identifies the 'ECA_INTR_STATE' for the interrupt logic.

bits	name	s/w	h/w	default	description
2	DONE	rw	ro	0x0	Done interrupt enable

					0: do not generate interrupt 1: generate interrupt if ECA_INT_STATE.DONE resetsignal : SW_Reset
1	ERR	rw	ro	0x1	Error interrupt enable 0: do not generate interrupt 1: generate interrupt if ECA_INT_STATE.ERR resetsignal : SW_Reset
0	ALERT	rw	ro	0x1	Alert interrupt enable 0: do not generate interrupt 1: generate interrupt if ECA_INT_STATE.ALERT resetsignal : SW_Reset

1.10.13 ECA_INTR_TEST

Reg. 0x000D4E68

General purpose interrupt test register with a bit to force each interrupt for test and debug.
rtl.reg_enb : false
display_name : ECA Interrupt Test

bits	name	s/w	h/w	default	description
2	DONE	rw	ro	0x0	Assert done interrupt 0: do not force interrupt 1: force ECA_INTR_STATE.DONE dontcompare : true resetsignal : SW_Reset
1	ERR	rw	ro	0x0	Assert error interrupt 0: do not force interrupt 1: force ECA_INTR_STATE.ERR dontcompare : true resetsignal : SW_Reset
0	ALERT	rw	ro	0x0	Assert alert interrupt 0: do not force interrupt 1: force ECA_INTR_STATE.ALERT dontcompare : true resetsignal : SW_Reset

1.10.14 ECA_MEM_TEST0

Reg. 0x000D4E6C

ECA Debug register.
rtl.reg_enb : false
display_name : DFT control and status

bits	name	s/w	h/w	default	description
24	MEM_TEST_ERR	r/w1c	wo	0x0	Memory test data in and out miscompares (fatal). Could indicate Bad memory or Bad memory connection sticky : true resetsignal : SW_Reset
19:8	MEM_TEST_ADDR	rw	ro	0x0	Memory byte address for testing memory connection. To test a memory line, Addr[11:4] = memory_line; addr[3:0] = 0x0; data = pattern_lsw addr[3:0] = 0x4; .. addr[3:0] = 0x8; .. addr[3:0] = 0xC; data = pattern_msw resetsignal : SW_Reset
4	MEM_TEST_EN	rw	ro	0x0	Control to enable memory testing. 0: Normal operation 1: Memory testmode resetsignal : SW_Reset
1	MEM_SERR_TEST	rw	ro	0x0	Memory single-bit error injection for test. When set, 1 wrong ECC bit will be written to the memory in the next write transactions. When read operation is performed for the relevant address, correctable error will be detected and corrected. Valid only when i_eca_debug_en==1. resetsignal : SW_Reset
0	MEM_DERR_TEST	rw	ro	0x0	Memory double-bit error injection for test. When set, 2 wrong ECC bits will be written to the memory in the next write transactions. When read operation is

					performed for the relevant address, ubcorrectable error will be detected. Valid only when i_eca_debug_en==1. resetsignal : SW_Reset

1.10.15 ECA_MEM_TEST1

0x000D4E70

Data for testing memory connection
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	MEM_TEST_DATA	rw	ro	0x0	Data input pattern for testing memory connection. Set RSA_MEM_TEST0.MEM_TEST_EN= 1 before using this register to test the memory. rtl.hw_wp : true

1.10.16 ECA_DEBUG

0x000D4E74

ECA Debug register, containing internal control signals
rtl.reg_enb : false
'ECA_DEBUG' is an external.
display_name : Debug register
no_reg_hw_reset_test : true

bits	name	s/w	h/w	default	description
29	INTR_STATE_DBG_DONE	ro	wo	0x0	Interrupt - Done
28	INTR_STATE_DBG_ERROR	ro	wo	0x0	Interrupt - Done
27	INTR_STATE_DBG_ALERT	ro	wo	0x0	Interrupt - Done
26:25	RD_BUFF_LINE_INDEX	ro	wo	0x0	Interrupt - Done
24:21	WR_BUFF_VALID_DIGIT	ro	wo	0x0	Interrupt - Done
20:19	DBG_CASE_SEL	ro	wo	0x0	Mult Inverse core current case select dontcompare : true
18	BUSY	ro	wo	0x0	ROT_ECA_INV Busy
17:16	INVERSE_CMD_CS	ro	wo	0x0	Mult Inverse FDM State
15:14	ADD_INV_CMD_CS	ro	wo	0x0	ADD Inverse FSM State
13:12	ADD_CMD_CS	ro	wo	0x0	ADD CMD FSM State
11:10	OP_CMD_CS	ro	wo	0x0	Main FSM State
9:8	DBG_ECA_MEM_CS	ro	wo	0x0	Inverse MEM Unit FSM State
7	MULT_BUSY	ro	wo	0x0	ROT_ECA_MULT Busy
6:5	MULT_MEM_CURRENT_STATE	ro	wo	0x0	Interrupt - Done
4:0	MULT_CURRENT_STATE	ro	wo	0x0	Interrupt - Done

End RegGroup

1.11 ROT_MAA

0x000D5000 - 0x000D5E2B

decode_size : 0x00001000
chapter : 1.5, Security Subsystem, MAA (RSA) Registers
module_name : rot_maa_regs
blockgroup : SECUREPROCESSOR
revision : revision: 02806f1

1.11.1 MAA_SEG_1

0x000D5000 - 0x000D51FF

Segment access.

rtl.reg_enb : false
'MAA_SEG_1' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write
'Count' : 'MAA_SEG_1' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5000	0xD5004	0xD5008	...	0xD51F4	0xD51F8	0xD51FC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.2 MAA_SEG_2

Reg.

0x000D5200 -
0x000D53FF

Segment access.
rtl.reg_enb : false
'MAA_SEG_2' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write
'Count' : 'MAA_SEG_2' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5200	0xD5204	0xD5208	...	0xD53F4	0xD53F8	0xD53FC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.3 MAA_SEG_3

Reg.

0x000D5400 -
0x000D55FF

Segment access.
rtl.reg_enb : false
'MAA_SEG_3' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write
'Count' : 'MAA_SEG_3' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5400	0xD5404	0xD5408	...	0xD55F4	0xD55F8	0xD55FC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.4 MAA_SEG_4

Reg.

0x000D5600 -
0x000D57FF

Segment access.
rtl.reg_enb : false
'MAA_SEG_4' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write
'Count' : 'MAA_SEG_4' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5600	0xD5604	0xD5608	...	0xD57F4	0xD57F8	0xD57FC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.5 MAA_SEG_5

Reg.

0x000D5800 -
0x000D59FF

Segment access.
rtl.reg_enb : false
'MAA_SEG_5' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write

'Count' : 'MAA_SEG_5' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5800	0xD5804	0xD5808	...	0xD59F4	0xD59F8	0xD59FC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.6 MAA_SEG_6

Reg. 0x000D5A00 - 0x000D5BFF

Segment access.
rtl.reg_enb : false
'MAA_SEG_6' is an external.
no_reg_tests : true
display_name : MAA Memory Read Write
'Count' : 'MAA_SEG_6' will repeat '128' times.

count	1	2	3	...	126	127	128
address	0xD5A00	0xD5A04	0xD5A08	...	0xD5BF4	0xD5BF8	0xD5BFC

bits	name	s/w	h/w	default	description
31:0	DATA_IN	rw	rw	0x0	segment memory access

1.11.7 MAA_CTRL

Reg. 0x000D5E00

General purpose control register
rtl.reg_enb : false
display_name : MAA Control

bits	name	s/w	h/w	default	description
13	X_CLEAR	wo	ro	0x0	Clear the X operand in memory. When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.X_READY. The operand needs to be written in memory again before an operation with it can be performed. resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'X_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.
12	E_CLEAR	wo	ro	0x0	Clear the E operand in memory. When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.E_READY. The operand needs to be written in memory again before an operation with it can be performed. resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'E_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.
11	N_CLEAR	wo	ro	0x0	Clear the N operand in memory. When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.N_READY. The operand needs to be written in memory again before an operation with it can be performed. resetsignal : SW_Reset 'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field. 'lock' : 'N_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.
10	R2MODN_CLEAR	wo	ro	0x0	Clear the R2MODN operand in memory. When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.R2MODN_READY. The operand needs to be written in memory again before an operation with it can be performed. resetsignal : SW_Reset

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
'lock' : 'R2MODN_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>9</td><td>RESULT_CLEAR</td><td>wo</td><td>ro</td><td>0x0</td><td>Clear the RESULT operand in memory.
When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.RESULT_READY.
The operand needs to be written in memory again before an operation with it can be performed.
resetsignal : SW_Reset
'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
'lock' : 'RESULT_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>8</td><td>D_CLEAR</td><td>wo</td><td>ro</td><td>0x0</td><td>Clear the D operand in memory.
When set, operand is marked as invalid in memory by clearing this operand ready flag in MAA_STATUS.D_READY.
The operand needs to be written in memory again before an operation with it can be performed.
resetsignal : SW_Reset
'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
'lock' : 'D_CLEAR' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>7:4</td><td>CMD</td><td>wo</td><td>ro</td><td>0x0</td><td>Control to indicate command for operation
(Use hamming encoded or one-hot values for following valid configurations)

enum:MAA_COMMAND_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>R2MODN</td><td>1</td><td>Pre-calculation R^2mod(n) operation</td></tr>
<tr><td>EXP1</td><td>2</td><td>Modular exponentiation x^emod(n) (full exponentiation)</td></tr>
<tr><td>EXP2</td><td>4</td><td>Modular exponentiation x^emod(n) using pre-calculated R^2mod(n)</td></tr>
<tr><td>RSVD</td><td>8</td><td>Reserved</td></tr>
</table>
encode : MAA_COMMAND_e
resetsignal : SW_Reset
'lock' : 'CMD' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>1</td><td>START</td><td>wo</td><td>ro</td><td>0x0</td><td>Control to indicate start of a new operation.
Software should write to this field to indicate core to start operation indicated in MAA_CTRL.CMD
resetsignal : SW_Reset
'singlepulse ' : It create a single cycle pulse on hardware interface and then in the next cycle it will clear the field.
'lock' : 'START' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>0</td><td>SW_RST</td><td>wo</td><td>ro</td><td>0x0</td><td>Active high soft reset
rtl.hw_wp : true</td></tr>
</table>

1.11.8 MAA_CFG

0x000D5E04

General purpose configuration register.
rtl.reg_enb : false
display_name : MAA Configuration

bits	name	s/w	h/w	default	description
11:8	OPD_SIZE	rw	ro	0x0	Size of the operand to use

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>(Use hamming encoded or one-hot values for following valid configurations)

enum:MAA_OPD_SIZE_e<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>OPD_2048</td><td>1</td><td>Use 2048-bit operands</td></tr><tr><td>OPD_3072</td><td>2</td><td>Use 3072-bit operands</td></tr><tr><td>OPD_4096</td><td>4</td><td>Use 4096-bit operands</td></tr><tr><td>OPD_RSVD</td><td>8</td><td>Reserved</td></tr></table>encode : MAA_OPD_SIZE_e
rtl.hw_wp : true
resetsignal : SW_Reset
'lock' : 'OPD_SIZE' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>5:4</td><td>EXP_TYPE</td><td>rw</td><td>ro</td><td>0x0</td><td>Exponent to use
(Use hamming encoded or one-hot values for following valid configurations)

enum:MAA_EXP_TYPE_e<table><tr><th>Name</th><th>Value</th><th>Description</th></tr><tr><td>PUB</td><td>1</td><td>Use public exponent E for operations in CMD</td></tr><tr><td>PRV</td><td>2</td><td>Use private exponent D for operations in CMD</td></tr></table>encode : MAA_EXP_TYPE_e
resetsignal : SW_Reset
'lock' : 'EXP_TYPE' is lock by 'MAA_STATUS.BUSY == 1'b1'.</td></tr>
<tr><td>1</td><td>BYTE_SWAP_IN</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable endian byte swapping for input data
Byte swapping based on this control applies to MAA memory segments writes
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
<tr><td>0</td><td>BYTE_SWAP_OUT</td><td>rw</td><td>ro</td><td>0x0</td><td>Enable endian byte swapping for output data
Byte swapping based on this control applies to MAA memory segments reads
0: Disable
1: Enable
resetsignal : SW_Reset</td></tr>
</table>

1.11.9 MAA_STATUS

Reg. 0x000D5E08

General purpose status register
rtl.reg_enb : false
display_name : MAA Status

bits	name	s/w	h/w	default	description
9	X_READY	ro	rw	0x0	Status of X operand in memory rtl.hw_clear : true resetsignal : SW_Reset
8	E_READY	ro	rw	0x0	Status of E operand in memory rtl.hw_clear : true resetsignal : SW_Reset
7	N_READY	ro	rw	0x0	Status of N operand in memory rtl.hw_clear : true resetsignal : SW_Reset
6	R2MODN_READY	ro	rw	0x0	Status of R2MODN operand in memory rtl.hw_clear : true resetsignal : SW_Reset
5	RESULT_READY	ro	wo	0x0	Status of RESULT operand in memory

					rtl.hw_clear : true resetsignal : SW_Reset we : true
4	D_READY	ro	rw	0x0	Status of D operand in memory rtl.hw_clear : true resetsignal : SW_Reset
1	BUSY	ro	rw	0x0	State of core 0: Idle (state == IDLE) 1: Busy (state != IDLE) resetsignal : SW_Reset
0	SW_RST_DONE	ro	wo	0x1	Software reset completion status 0: Reset not successful 1: Reset successful rtl.hw_set : true rtl.hw_clear : true we : true

1.11.10 MAA_ERR_STATUS

Reg. 0x000D5E0C

General error status register with status bit for each error event
rtl.reg_enb : false
display_name : MAA Error Status

bits	name	s/w	h/w	default	description
21	MAA_SERR	r/w1c	rw	0x0	Single bit error detected and corrected by SecDed (non-fatal) resetsignal : SW_Reset
20	MAA_DERR	ro	rw	0x0	Double bit error detected by SecDed (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. resetsignal : SW_Reset
16	R2_ERR	ro	wo	0x0	Invalid R2 operand write - trying to override R2 withoud clearing its ready. This error causes The core to cease all operation and can only be cleared by a reset before re-use. * R2_ERR[0]: R2 overwrite detected. resetsignal : SW_Reset we : true
14:12	X_ERR	ro	wo	0x0	Invalid X value or length detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * X_ERR[2]: X >= N * X_ERR[1]: X <= 1 * X_ERR[0]: X overwrite detected. resetsignal : SW_Reset we : true
10:8	E_ERR	ro	wo	0x0	Invalid E value or length detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * E_ERR[2]: E >= N * E_ERR[1]: E <= 1 * E_ERR[0]: E overwrite detected. resetsignal : SW_Reset we : true
6:4	N_ERR	ro	wo	0x0	Invalid N value or length detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * N_ERR[2]: N is even * N_ERR[1]: N has more than 31 zeros in MSD * N_ERR[0]: N overwrite detected. resetsignal : SW_Reset we : true

2:0	D_ERR	ro	wo	0x0	Invalid D value or length detected (fatal). This error causes The core to cease all operation and can only be cleared by a reset before re-use. * D_ERR[2]: Length(d) > MAA_CFG.OPD_SIZE * D_ERR[1]: D <= 1 * D_ERR[0]: D overwrite detected. resetsignal : SW_Reset we : true

1.11.11 MAA_ALERT_STATUS

Reg. 0x000D5E10

Security relevant error status register with a status bit for each alert event.
rtl.reg_enb : false
display_name : MAA Alert Status

bits	name	s/w	h/w	default	description
2	EXP_TYPE_ALERT	ro	wo	0x0	An invalid exponent type was detected (fatal). If value of MAA_CFG.EXP_TYPE is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset we : true
1	MOD_SIZE_ALERT	ro	wo	0x0	An invalid modulus size was detected (fatal). If value of MAA_CFG.MOD_SIZE is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset we : true
0	CMD_ALERT	ro	wo	0x0	An invalid command was detected (fatal). If value of MAA_CTRL.CMD is outside of the legal coded values, the core should set this flag. resetsignal : SW_Reset we : true

1.11.12 MAA_INTR_STATE

Reg. 0x000D5E14

General purpose interrupt status register with each bit corresponding to an interrupt port. There is 1 bit for consolidated alert and error events each, and a dedicated bit for any other status events that need to be routed as interrupts. All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be re-latched.
rtl.reg_enb : false
display_name : MAA Interrupt State

bits	name	s/w	h/w	default	description
2	DONE	r/w1c	rw	0x0	Command done interrupt. Status indicating current operation is done, and results are ready to be read. resetsignal : SW_Reset
1	ERR	r/w1c	rw	0x0	Error interrupt. Status indicating an error has been detected. This bit is a consolidation (OR) of all error events in MAA_ERR_STATUS. resetsignal : SW_Reset
0	ALERT	r/w1c	rw	0x0	Alert interrupt. A security relevant error has been detected. This bit is a consolidation (OR) of all alert events in MAA_ALERT_STATUS.. resetsignal : SW_Reset

1.11.13 MAA_INTR_ENABLE

Reg. 0x000D5E18

General purpose interrupt enable status register with a bit to mask/unmask each interrupt port.
rtl.reg_enb : false
display_name : MAA Interrupt Enable

'intr.mask' : Identifies the 'MAA_INTR_STATE' for the interrupt logic.

bits	name	s/w	h/w	default	description
2	DONE	rw	ro	0x0	Done interrupt enable 0: do not generate interrupt 1: generate interrupt if MAA_INT_STATE.DONE resetsignal : SW_Reset
1	ERR	rw	ro	0x1	Error interrupt enable 0: do not generate interrupt 1: generate interrupt if MAA_INT_STATE.ERR resetsignal : SW_Reset
0	ALERT	rw	ro	0x1	Alert interrupt enable 0: do not generate interrupt 1: generate interrupt if MAA_INT_STATE.ALERT resetsignal : SW_Reset

1.11.14 MAA_INTR_TEST

Reg. 0x000D5E1C

General purpose interrupt test register with a bit to force each interrupt for test and debug.
rtl.reg_enb : false
dontcompare : true
display_name : MAA Interrupt Test

bits	name	s/w	h/w	default	description
2	DONE	rw	ro	0x0	Assert done interrupt 0: do not force interrupt 1: force MAA_INTR_STATE.DONE dontcompare : true resetsignal : SW_Reset
1	ERR	rw	ro	0x0	Assert error interrupt 0: do not force interrupt 1: force MAA_INTR_STATE.ERR dontcompare : true resetsignal : SW_Reset
0	ALERT	rw	ro	0x0	Assert alert interrupt 0: do not force interrupt 1: force MAA_INTR_STATE.ALERT dontcompare : true resetsignal : SW_Reset

1.11.15 MAA_MEM_TEST0

Reg. 0x000D5E20

MAA Debug register.
rtl.reg_enb : false
display_name : DFT control status

bits	name	s/w	h/w	default	description
24	MEM_TEST_ERR	r/w1c	wo	0x0	Memory test data in and out miscompares (fatal). Could indicate Bad memory or Bad memory connection sticky : true resetsignal : SW_Reset
19:8	MEM_TEST_ADDR	rw	ro	0x0	Memory byte address for testing memory connection. To test a memory line, Addr[11:4] = memory_line; addr[3:0] = 0x0; data = pattern_lsw addr[3:0] = 0x4; .. addr[3:0] = 0x8; .. addr[3:0] = 0xC; data = pattern_msw resetsignal : SW_Reset
4	MEM_TEST_EN	rw	ro	0x0	Control to enable memory testing. 0: Normal operation 1: Memory testmode resetsignal : SW_Reset
1	MEM_SERR_TEST	rw	ro	0x0	Memory single-bit error injection for test. When set, 1 wrong ECC bit will be written to the memory in the next write transactions. When read operation is performed for the relevant address, correctable error will be detected and corrected. Valid only when i_maa_debug_en==1.

					resetsignal : SW_Reset
0	MEM_DERR_TEST	rw	ro	0x0	Memory double-bit error injection for test. When set, 2 wrong ECC bits will be written to the memory in the next write transactions. When read operation is performed for the relevant address, ubcorrectable error will be detected. Valid only when i_maa_debug_en==1. resetsignal : SW_Reset

1.11.16 MAA_MEM_TEST1

0x000D5E24

Data for testing memory connection
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	MEM_TEST_DATA	rw	ro	0x0	Data input pattern for testing memory connection. Set RSA_MEM_TEST0.MEM_TEST_EN= 1 before using this register to test the memory. rtl.hw_wp : true resetsignal : SW_Reset

1.11.17 MAA_DEBUG

0x000D5E28

MAA Debug register, containing internal control signals
rtl.reg_enb : false
'MAA_DEBUG' is an external.
display_name : Debug register

bits	name	s/w	h/w	default	description
31	INTR_STATE_DBG_DONE	ro	wo	0x0	Interrupt - Done
30	INTR_STATE_DBG_ERROR	ro	wo	0x0	Interrupt - Done
29	INTR_STATE_DBG_ALERT	ro	wo	0x0	Interrupt - Done
28:22	RD_WR_DATA_CNT	ro	wo	0x0	Read/Write to memory Counter
21:19	ZMN_CURRENT_STATE	ro	wo	0x0	ZMN FSM state
18:16	DELTA_CURRENT_STATE	ro	wo	0x0	DELTA FSM state
15:12	MEM_IF_CURRENT_STATE	ro	wo	0x0	MEM_IF FSM state
11:7	ARITH_CURRENT_STATE	ro	wo	0x1	ARITH FSM state
6:4	MM_CURRENT_STATE	ro	wo	0x0	MM FSM state
3:0	MAIN_CURRENT_STATE	ro	wo	0x0	MAIN FSM state

End RegGroup

1.12 RSA_REGS_SYS

0x000D6000 - 0x000D607B

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00001000
chapter : 1.31, Security Subsystem, CS RSA
blockgroup : SECUREPROCESSOR
revision : revision: 872aeff

1.12.1 rsa_regs

0x000D6000 - 0x000D607B

1.12.1.1 RSA_DATA_FOR_MEMORY

0x000D6000

bits	name	s/w	h/w	default	description
31:0	WDATA	rw	rw	0x0	

1.12.1.2 RSA_MEMORY_WRITE_ADDRESS

0x000D6008

bits	name	s/w	h/w	default	description
7:0	ADDR	wo	rw	0x0	
31	WRITE_EN	wo	rw	0x0	

1.12.1.3 RSA_CONTROL

0x000D6010

bits	name	s/w	h/w	default	description
2:0	START_CODE	wo	rw	0x0	
6	CLEAR	wo	rw	0x0	
7	CLEAR_BUF	wo	rw	0x0	
8	CLR_MEM_CORERR	wo	rw	0x0	
9	CLR_MEM_UNCERR	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.4 RSA_STATUS

0x000D6018

bits	name	s/w	h/w	default	description
0	DONE	ro	rw	0x0	
1	BUSY	ro	rw	0x0	
2	ERROR	ro	rw	0x0	
7:3	PROGRESS	ro	rw	0x0	
8	COR_MEM_ERR	ro	rw	0x0	
9	UNCOR_MEM_ERR	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.5 RSA_MINV

0x000D6020

bits	name	s/w	h/w	default	description
31:0	MINV	wo	rw	0x0	

1.12.1.6 RSA_INTERRUPT_ENABLE

0x000D6028

bits	name	s/w	h/w	default	description
0	DONE	rw	rw	0x0	
1	MEM_ERR	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.7 RSA_CEN_CONTROL

0x000D6030

bits	name	s/w	h/w	default	description
0	CEN	rw	rw	0x1	
1	COR_ERR_INJ	rw	rw	0x0	
2	UNCOR_ERR_INJ	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.8 RSA_LENGTH

Reg. 0x000D6040

bits	name	s/w	h/w	default	description
14:0	LENGTH	wo	rw	0x0	
31	MODE	wo	rw	0x0	

1.12.1.9 RSA_DATA_LENGTH

Reg. 0x000D6050

bits	name	s/w	h/w	default	description
5:0	MULTIPLICATION	wo	rw	0x0	
7:6	FINAL_QSEL	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.10 RSA_MONTGOMERY_DATA

Reg. 0x000D6058

bits	name	s/w	h/w	default	description
7:0	MONTGOMERY	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.11 RSA_MEMORY_SIZE

Reg. 0x000D6060

bits	name	s/w	h/w	default	description
4:0	LOCATION_SIZE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.12 RSA_MULTIPLICATION

Reg. 0x000D6068

bits	name	s/w	h/w	default	description
3:0	SCRATCH_LOC	wo	rw	0x0	
7:4	MONTGOMERY_LOC	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.12.1.13 RSA_MICROPROGRAM

Reg. 0x000D6070

bits	name	s/w	h/w	default	description
31:0	UNPROG_START_ADDRESS	wo	rw	0x0	

1.12.1.14 RSA_GROUPS

Reg. 0x000D6078

bits	name	s/w	h/w	default	description
4:0	GROUPS	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

End RegGroup

End RegGroup

1.13 ROT_TRNG

RegGrp 0x000D7000 - 0x000D7447

Created by Broadcom Register Toolkit [Version: 2019.4 - a3ba04c6d5636afb09cfd703888d7e1c5c1bd4aa] on 2019-7-5 21:06:53 UTC
decode_size : 0x00000800
chapter : 1.6, Security Subsystem, ROT TRNG SECR_TRNG_800_90_AB Registers
blockgroup : SECUREPROCESSOR
revision : revision: 4c3e05f

1.13.1 TRNG_RESERVED

Reg. 0x000D7000

bits	name	s/w	h/w	default	description
31:0	UNUSED	ro	na	0x0	

1.13.2 TRNG_SCRATCH

Reg. 0x000D7004

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0x0	Software scratch register. Not used by hardware

1.13.3 TRNG_CMD

Reg. 0x000D7010

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:4	TRNG_cmd_spare0	rw	ro	0x0	Reserved
3:0	cmd	rw	ro	0x0	The register is used to issue the following commands: 001 - INSTANTIATE, 010 - RESEED, 011 - GENERATE, 100 - TEST, 101 - UNINSTANTIATE, 110 - STANDBY 111 - RESET UNINSTANTIATE command is accepted at all times, other commands are only accepted if trng_sts_cmd_rdy is 1. dontcompare : true

1.13.4 TRNG_STS

Reg. 0x000D7020

bits	name	s/w	h/w	default	description
31:24	data_cnt	ro	wo	0x0	Number of 32bit words ready to be read
23:20	TRNG_sts_spare1	ro	wo	0x0	Reserved.
19	es_adprop_bist_sts	ro	wo	0x0	ES Adaptive Proportion test bist status, 1=expected to fail during esbist but passed
18	es_repcnt_bist_sts	ro	wo	0x0	ES Repitition Count test bist status, 1=expected to fail during esbist but passed
17	es_longrun_bist_sts	ro	wo	0x0	ES longrun test bist status, 1=expected to fail during esbist but passed
16	reseed_required	ro	wo	0x0	1 if reseed is required
15	kat_sts	ro	wo	0x0	TRNG KAT status, 1=fail
14	cont_test_sts	ro	wo	0x0	TRNG ES Continuous test status, 1=fail. es_adprop_test_sts,es_repcnt_sts, es_longrun_sts show which test failed
13	startup_test_sts	ro	wo	0x0	TRNG Startup test status, 1=fail. Startup tests are run after TRNG BIST tests pass
12	bist_sts	ro	wo	0x0	TRNG BIST test status, 1=fail. TRNG BIST tests are run upon reset and test commands
11	es_adprop_test_sts	ro	wo	0x0	ES Adaptive Proportion test status, 1=fail
10	es_repcnt_test_sts	ro	wo	0x0	ES Repitition Count test status, 1=fail
9	es_longrun_test_sts	ro	wo	0x0	ES longrun status, 1=fail

8	es_bist_sts	ro	wo	0x0	ES Bist test status, 1=fail
7:6	TRNG_sts_spare0	ro	wo	0x0	Reserved.
5	cmd_err	ro	wo	0x0	If 1 trng cmd has error
4	cmd_rdy	ro	wo	0x1	If 1 trng cmd is done
3:0	cmd	ro	wo	0x0	The last command that was accepted.

1.13.5 TRNG_POOL_STS

Reg. 0x000D7030

bits	name	s/w	h/w	default	description
31:0	rosc_longrun_sts	ro	wo	0x0	Bit is 0 when corresponding ring oscillator is OK and 1 when ring oscillator generated the same value during 48 clock cycles.

1.13.6 TRNG_DOUT

Reg. 0x000D7040

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Generated random word.

1.13.7 TRNG_ROSC_OUT

Reg. 0x000D7050

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Output from ring oscillators pool. Disabled after TRNG is instantiated

1.13.8 TRNG_SAMPLED_ROSC_OUT

Reg. 0x000D7054

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Sampled ring oscillators pool data. Disabled after TRNG is instantiated

1.13.9 TRNG_SAMPLED_OVERFLOW_CNT

Reg. 0x000D7058

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Count of number of times overflow occurred for sampled ring oscillator data, i.e. data was updated before it was read out over APB

1.13.10 TRNG_REQ_LEN

Reg. 0x000D7060

bits	name	s/w	h/w	default	description
31:12	req_len_spare0	ro	ro	0x0	Reserved.
11:0	val	rw	ro	0x0	Number of 128b blocks to be generated

1.13.11 TRNG_USER_IN_LEN

Reg. 0x000D7070

bits	name	s/w	h/w	default	description
31:6	user_in_len_spare0	ro	ro	0x0	Reserved.
5:0	val	rw	ro	0x0	Bytelength of user input

1.13.12 TRNG_CTRL

Reg. 0x000D7080

bits	name	s/w	h/w	default	description
31:3	trng_ctrl_spare0	ro	ro	0x0	Reserved.
2	enable_longrun	rw	ro	0x0	If 1 ES longrun test is enabled dontcompare : true
1	prediction_resistance_on	rw	ro	0x0	If 1 prediction resistance is enabled dontcompare : true
0	use_ro	rw	ro	0x1	If 1 ring oscillators' pool is used as entropy input and nonce, otherwise use entropy input and nonce register dontcompare : true

1.13.13 TRNG_ROSC_CTRL

Reg. 0x000D7084

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:22	rosc_ctrl_spare3	ro	ro	0x0	Reserved
21:16	rosc_ctrl_hr	ro	ro	0x4	HR (unused)
15:0	rosc_ctrl_sr	rw	ro	0x64	SR.This bit field must be adjusted only on direction from Broadcom

1.13.14 TRNG_ROSC_SELFTEST_CTRL

Reg. 0x000D7088

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0xFFFFFFFF	rosc_selftest_ctrl.This bit field must be adjusted only on direction from Broadcom

1.13.15 TRNG_MIN_ENTR

Reg. 0x000D7090

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:5	trng_min_entr_spare0	ro	ro	0x0	Reserved
4:0	val	rw	ro	0x4	Assessed min-entropy. This bit field must be adjusted only on direction from Broadcom

1.13.16 TRNG_ENTROPY

Reg. 0x000D7100 - 0x000D711F

'Count' : 'TRNG_ENTROPY' will repeat '8' times.

count	1	2	3	...	6	7	8
address	0xD7100	0xD7104	0xD7108	...	0xD7114	0xD7118	0xD711C

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0x0	Word i of entropy input provided by user

1.13.17 TRNG_NONCE

Reg. 0x000D7120 - 0x000D712F

'Count' : 'TRNG_NONCE' will repeat '4' times.

count	0	1	2	3
address	0xD7120	0xD7124	0xD7128	0xD712C

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0x0	Word i of nonce input provided by user for instantiate command

1.13.18 TRNG_USER_IN

0x000D7200 -
0x000D722F

'Count' : 'TRNG_USER_IN' will repeat '12' times.

count	1	2	3	...	10	11	12
address	0xD7200	0xD7204	0xD7208	...	0xD7224	0xD7228	0xD722C

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0x0	Word i of user input provided by user

1.13.19 TRNG_ISR

0x000D7300

bits	name	s/w	h/w	default	description
31:2	trng_isr_spare0	r/w1c	ro	0x0	Reserved
1	data_vld	r/w1c	rw	0x0	Data valid interrupt Random numbers are ready to be read we : true
0	trng_err	r/w1c	rw	0x0	TRNG error interrupt TRNG error occurred, additional information can be found in sts register Reset (hard or soft) is needed we : true

1.13.20 TRNG_IER

0x000D7304

bits	name	s/w	h/w	default	description
31:2	trng_ier_spare0	ro	ro	0x0	Reserved
1	data_vld	rw	ro	0x0	Enable 'data valid' interrupt dontcompare : true
0	trng_err	rw	ro	0x0	Enable 'trng error' interrupt dontcompare : true

1.13.21 TRNG_IFR

0x000D7308

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:2	trng_ifr_spare0	ro	ro	0x0	Reserved
1	data_vld	rw	ro	0x0	Force 'data valid' interrupt for testing (if enabled)
0	trng_err	rw	ro	0x0	Force 'trng error' interrupt for testing (if enabled)

1.13.22 TRNG_CHAR_SCRATCH

0x000D7400

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0x0	Software scratch register. Not used by hardware

1.13.23 TRNG_CHAR_CMD

0x000D7404

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:4	char_cmd_spare0	rw	ro	0x0	Reserved
3:0	cmd	rw	ro	0x0	The register is used to issue the following commands: 001 - INSTANTIATE, 010 - RESEED, 011 - GENERATE, 100 - TEST, 101 - UNINSTANTIATE, 110 - STANDBY 111 - RESET

1.13.24 TRNG_CHAR_STS

0x000D7408

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:20	char_sts_spare2	ro	wo	0x0	Reserved
19	es_adprop_bist_sts	ro	wo	0x0	ES Adaptive Proportion test bist status, 1=expected to fail during esbist but passed
18	es_repcnt_bist_sts	ro	wo	0x0	ES Repitition Count test bist status, 1=expected to fail during esbist but passed
17	es_longrun_bist_sts	ro	wo	0x0	ES longrun test bist status, 1=expected to fail during esbist but passed
16:15	char_sts_spare1	ro	wo	0x0	Reserved
14	cont_test_sts	ro	wo	0x0	TRNG Continuous test status, 1=fail
13	startup_test_sts	ro	wo	0x0	TRNG Startup test status, 1=fail
12	bist_sts	ro	wo	0x0	TRNG BIST test status, 1=fail
11	es_adprop_test_sts	ro	wo	0x0	ES Adaptive Proportion test status, 1=fail
10	es_repcnt_test_sts	ro	wo	0x0	ES Repitition Count test status, 1=fail
9	es_longrun_test_sts	ro	wo	0x0	ES longrun status, 1=fail
8	es_bist_sts	ro	wo	0x0	ES Bist test status, 1=fail
7:6	char_sts_spare0	ro	wo	0x0	Reserved.
5	cmd_err	ro	wo	0x0	If 1 trng cmd has error
4	cmd_rdy	ro	wo	0x1	If 1 trng cmd is done
3:0	cmd	ro	wo	0x0	The last command that was accepted

1.13.25 TRNG_CHAR_POOL_STS

0x000D740C

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:0	rosc_longrun_sts	ro	wo	0x0	Bit is 0 when correspondent ring oscillator is OK and 1 when ring oscillator generated the same value during 48 clock cycles

1.13.26 TRNG_CHAR_ROSC_CTRL

0x000D7410

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:29	char_rosc_ctrl_unused2	ro	ro	0x0	Unused2
28	rosc_alpha30	rw	ro	0x0	Use alpha=2^-30 for repcnt and adprop tests
27	rosc_disable_rosc	rw	ro	0x0	Disable rosc(use lfsr instead)
26	rosc_disable_repcnt	rw	ro	0x0	Disable repcnt test
25	rosc_disable_adprop	rw	ro	0x0	Disable adprop test
24	rosc_disable_longrun	rw	ro	0x0	Disable longrun test
23:22	char_rosc_ctrl_unused1	ro	ro	0x0	Unused1
21:16	rosc_ctrl_hr	rw	ro	0x0	Rosc HR
15:0	rosc_ctrl_sr	rw	ro	0x0	Rosc SR

1.13.27 TRNG_CHAR_MIN_ENTR

0x000D7414

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:5	char_min_entr_spare0	ro	ro	0x0	Reserved

4:0	val	rw	ro	0x4	Minimum entropy

1.13.28 TRNG_CHAR_ROSC_OUT

0x000D7418

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0xFFFFFFFF	Output from char ring oscillators pool

1.13.29 TRNG_CHAR_SAMPLED_ROSC_OUT

0x000D741C

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Sampled output from char ring oscillators pool

1.13.30 TRNG_CHAR_SAMPLED_OVERFLOW_CNT

0x000D7420

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:0	val	ro	wo	0x0	Count of number of times overflow occurred for sampled ring oscillator data, i.e. data was updated before it was read out over APB

1.13.31 TRNG_CHAR_ROSC_SELFTEST_CTRL

0x000D7424

no_reg_tests : true

bits	name	s/w	h/w	default	description
31:0	val	rw	ro	0xFFFFFFFF	rosc_selftest_ctrl.This bit field must be adjusted only on direction from Broadcom

1.13.32 TRNG_CS_CTRL

0x000D7430

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:16	cs_post_trigger_count	rw	ro	0x0	Post Trigger Count, CS will capture these additional samples post trigger
15:14	cs_spare2	rw	ro	0x0	Reserved
13:8	cs_strobe_select	rw	ro	0x0	Index of bit # in TriggerSignals to use as strobe
7:6	cs_ctrl_spare1	rw	ro	0x0	Reserved
5:4	cs_capture_select	rw	ro	0x0	Select signal to capture: 00 : xor_osc_out 01 : xor_osc_out 10 : sync_osc_out 11 : raw_osc_out
3	cs_ctrl_spare0	rw	ro	0x0	Reserved
2	cs_abort	rw	ro	0x0	Abort current sequence
1	cs_acquire	rw	ro	0x0	Start acquire
0	cs_enable	rw	ro	0x0	Enable Chipscope

1.13.33 TRNG_CS_TRIGSEL

0x000D7434

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:0	cs_trigsel	rw	ro	0x0	Enable for match to trigger; If bit is 1 then corresponding bit is used to match for trigger, don’t care otherwise

1.13.34 TRNG_CS_TRIGPOL

Reg. 0x000D7438

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:0	cs_trigpol	rw	ro	0x0	Program polarity of signal for match to trigger; If bit is 1 then 1 on corresponding bit is used to match for trigger, 0 otherwise

1.13.35 TRNG_CS_ACQ_ADDR

Reg. 0x000D743C

no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
31:16	cs_acq_addr_spare0	rw	ro	0x0	Reserved
15:0	cs_acq_addr	rw	ro	0x0	Byte Address to query acquired data from chipscope; to be used after CS is triggered

1.13.36 TRNG_CS_ACQ_DATA

Reg. 0x000D7440

bits	name	s/w	h/w	default	description
31:16	cs_acq_data_spare0	ro	ro	0x0	Reserved
15:0	cs_acq_data	ro	wo	0x0	Halfword readback from CS for address specified in Acquired Address

1.13.37 TRNG_CS_CURR_ADDR

Reg. 0x000D7444

bits	name	s/w	h/w	default	description
31:16	cs_curr_addr	ro	wo	0x0	The Current address where the cs memory is last written post trigger. This address points to location equal to trigger location + PostTriggerCount
15:2	cs_curr_addr_spare0	ro	wo	0x0	Reserved
1	cs_triggered	ro	wo	0x0	Indicates that chipscope is triggered
0	cs_armed	ro	wo	0x0	Indicates that chipscope is armed and waiting for trogger

End RegGroup

1.14 DWC_TRNG_CORE

RegGrp 0x000D8000 - 0x000D80F7

Present for CSSD and CS SoCs only. Access write-ignored, read zeros for HDD/ESSD
decode_size : 0x00000800
chapter : 1.30, Security Subsystem, CS DW TRNG
blockgroup : SECUREPROCESSOR
revision : revision: 872aeff

1.14.1 DWC_trng_core_nist_trng_controller

RegGrp 0x000D8000 - 0x000D80F7

1.14.1.1 CTRL

Reg. 0x000D8000

bits	name	s/w	h/w	default	description

3:0	CMD	wo	rw	0x0	volatile : 1
31:4	Reserved_4_31	ro	rw	0x0	

1.14.1.2 MODE

Reg. 0x000D8004

bits	name	s/w	h/w	default	description
0	SEC_ALG	rw	rw	0x1	volatile : 1
2:1	Reserved_1_2	ro	rw	0x0	
3	PRED_RESIST	rw	rw	0x0	volatile : 1
4	ADDIN_PRESENT	rw	rw	0x0	volatile : 1
6:5	KAT_VEC	rw	rw	0x3	volatile : 1
8:7	KAT_SEL	rw	rw	0x2	volatile : 1
31:9	Reserved_9_31	ro	rw	0x0	

1.14.1.3 SMODE

Reg. 0x000D8008

bits	name	s/w	h/w	default	description
0	NONCE	rw	rw	0x0	volatile : 1
1	MISSION_MODE	rw	rw	0x1	volatile : 1
9:2	MAX_REJECTS	rw	rw	0xA	volatile : 1
15:10	Reserved_10_15	ro	rw	0x0	
23:16	INDIV_HT_DISABLE	rw	rw	0x0	volatile : 1
30:24	Reserved_24_30	ro	rw	0x0	
31	NOISE_COLLECT	rw	rw	0x0	volatile : 1

1.14.1.4 STAT

Reg. 0x000D800C

bits	name	s/w	h/w	default	description
3:0	LAST_CMD	ro	rw	0x0	volatile : 1
4	SEC_ALG	ro	rw	0x1	volatile : 1
5	NONCE_MODE	ro	rw	0x0	volatile : 1
6	MISSION_MODE	ro	rw	0x1	volatile : 1
8:7	DRBG_STATE	ro	rw	0x0	volatile : 1
9	STARTUP_TEST_STUCK	ro	rw	0x0	volatile : 1
10	STARTUP_TEST_IN_PROG	ro	rw	0x1	volatile : 1
30:11	Reserved_11_30	ro	rw	0x0	
31	BUSY	ro	rw	0x0	volatile : 1

1.14.1.5 IE

Reg. 0x000D8010

bits	name	s/w	h/w	default	description
0	ZEROIZED	rw	rw	0x0	
1	KAT_COMPLETED	rw	rw	0x0	
2	NOISE_RDY	rw	rw	0x0	
3	ALARMS	rw	rw	0x0	
4	DONE	rw	rw	0x0	
30:5	Reserved_5_30	ro	rw	0x0	
31	GLBL	rw	rw	0x0	

1.14.1.6 ISTAT

Reg. 0x000D8014

bits	name	s/w	h/w	default	description
0	ZEROIZED	r/w1c	rw	0x0	volatile : 1
1	KAT_COMPLETED	r/w1c	rw	0x0	volatile : 1
2	NOISE_RDY	r/w1c	rw	0x0	volatile : 1

3	ALARMS	r/w1c	rw	0x0	volatile : 1
4	DONE	r/w1c	rw	0x0	volatile : 1
31:5	Reserved_5_31	ro	rw	0x0	

1.14.1.7 ALARMS

Reg. 0x000D8018

bits	name	s/w	h/w	default	description
3:0	FAILED_TEST_ID	rw	rw	0x0	volatile : 1
4	ILLEGAL_CMD_SEQ	rw	rw	0x0	volatile : 1
5	FAILED_SEED_ST_HT	rw	rw	0x0	volatile : 1
31:6	Reserved_6_31	ro	rw	0x0	

1.14.1.8 COREKIT_REL

Reg. 0x000D801C

bits	name	s/w	h/w	default	description
15:0	REL_NUM	ro	rw	0x300B	
23:16	EXT_VER	ro	rw	0x0	
27:24	Reserved_24_27	ro	rw	0x0	
31:28	EXT_ENUM	ro	rw	0x0	

1.14.1.9 FEATURES

Reg. 0x000D8020

bits	name	s/w	h/w	default	description
0	SECURE_RST_STATE	ro	rw	0x1	
3:1	DIAG_LEVEL_ST_HLT	ro	rw	0x0	
6:4	DIAG_LEVEL_CLP800	ro	rw	0x0	
7	DIAG_LEVEL_NS	ro	rw	0x0	
8	PS_PRESENT	ro	rw	0x1	
9	AES_256	ro	rw	0x1	
31:10	Reserved_10_31	ro	rw	0x0	

1.14.1.10 RAND0

Reg. 0x000D8024

bits	name	s/w	h/w	default	description
31:0	RAND	ro	rw	0x0	volatile : 1

1.14.1.11 RAND1

Reg. 0x000D8028

bits	name	s/w	h/w	default	description
31:0	RAND	ro	rw	0x0	volatile : 1

1.14.1.12 RAND2

Reg. 0x000D802C

bits	name	s/w	h/w	default	description
31:0	RAND	ro	rw	0x0	volatile : 1

1.14.1.13 RAND3

Reg. 0x000D8030

bits	name	s/w	h/w	default	description
31:0	RAND	ro	rw	0x0	volatile : 1

1.14.1.14 NPA_DATA0

Reg. 0x000D8034

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.15 NPA_DATA1

Reg. 0x000D8038

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.16 NPA_DATA2

Reg. 0x000D803C

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.17 NPA_DATA3

Reg. 0x000D8040

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.18 NPA_DATA4

Reg. 0x000D8044

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.19 NPA_DATA5

Reg. 0x000D8048

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.20 NPA_DATA6

Reg. 0x000D804C

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.21 NPA_DATA7

Reg. 0x000D8050

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.22 NPA_DATA8

Reg. 0x000D8054

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.23 NPA_DATA9

Reg. 0x000D8058

bits	name	s/w	h/w	default	description

31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.24 NPA_DATA10

Reg. 0x000D805C

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.25 NPA_DATA11

Reg. 0x000D8060

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.26 NPA_DATA12

Reg. 0x000D8064

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.27 NPA_DATA13

Reg. 0x000D8068

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.28 NPA_DATA14

Reg. 0x000D806C

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.29 NPA_DATA15

Reg. 0x000D8070

bits	name	s/w	h/w	default	description
31:0	NPA_DATA	rw	rw	0x0	volatile : 1

1.14.1.30 SEED0

Reg. 0x000D8074

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.31 SEED1

Reg. 0x000D8078

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.32 SEED2

Reg. 0x000D807C

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.33 SEED3

Reg. 0x000D8080

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.34 SEED4

0x000D8084

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.35 SEED5

0x000D8088

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.36 SEED6

0x000D808C

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.37 SEED7

0x000D8090

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.38 SEED8

0x000D8094

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.39 SEED9

0x000D8098

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.40 SEED10

0x000D809C

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.41 SEED11

0x000D80A0

bits	name	s/w	h/w	default	description
31:0	SEED	rw	rw	0x0	volatile : 1

1.14.1.42 TIME_TO_SEED

0x000D80D0

bits	name	s/w	h/w	default	description
31:0	TTS	ro	rw	0x0	

1.14.1.43 BUILD_CFG0

0x000D80F0

bits	name	s/w	h/w	default	description
1:0	CORE_TYPE	ro	rw	0x2	
3:2	DIGITIZER_TYPE	ro	rw	0x1	
6:4	DIGITIZER_CNTR_WIDTH	ro	rw	0x4	
7	BG8	ro	rw	0x1	
9:8	CDC_SYNC_DEPTH	ro	rw	0x2	
10	BACKGROUND_NOISE	ro	rw	0x1	
11	EDU_PRESENT	ro	rw	0x0	
12	AES_DATAPATH	ro	rw	0x1	
13	AES_MAX_KEY_SIZE	ro	rw	0x1	
14	PERSONALIZATION_STR	ro	rw	0x1	
15	Reserved_15_15	ro	rw	0x0	
18:16	DIAGNOSTIC_LEVEL	ro	rw	0x0	
19	NS_DIAGNOSTIC_LEVEL	ro	rw	0x0	
31:20	Reserved_20_31	ro	rw	0x0	

1.14.1.44 BUILD_CFG1

Reg. 0x000D80F4

bits	name	s/w	h/w	default	description
7:0	NUM_RAW_NOISE_BLKS	ro	rw	0x4	
8	STICKY_STARTUP	ro	rw	0x1	
11:9	Reserved_9_11	ro	rw	0x0	
12	AUTO_CORRELATION_TEST	ro	rw	0x0	
13	MONOBIT_TEST	ro	rw	0x0	
14	RUN_TEST	ro	rw	0x0	
15	POKER_TEST	ro	rw	0x0	
18:16	RAW_HT_ADAP_TEST	ro	rw	0x1	
19	RAW_HT_REP_TEST	ro	rw	0x1	
22:20	ENT_SRC_REP_SMPL_SIZE	ro	rw	0x0	
23	ENT_SRC_REP_TEST	ro	rw	0x1	
30:24	ENT_SRC_REP_MIN_ENTROPY	ro	rw	0x50	
31	Reserved_31_31	ro	rw	0x0	

End RegGroup

End RegGroup

1.15 ROT_RNG

RegGrp 0x000D8800 - 0x000D8907

decode_size : 0x00000800
chapter : 1.62, Security Subsystem, ROT WD RNG Registers
module_name : rot_rng_regs
blockgroup : SECUREPROCESSOR
revision : revision: e8c4ddb

1.15.1 RNG_ETS_CTRL

Reg. 0x000D8800

General purpose control register
rtl.reg_enb : false
display_name : RNG ETS Control

bits	name	s/w	h/w	default	description
15:4	TRNG_TOP_EN	rw	ro	0x0	Enable per TRNG lane. '0': Disable '1': Enable resetsignal : SW_Reset
2	ES_XOR_SOURCE	rw	ro	0x0	Control to selects the source of Entropt-source XOR data. '0': Data source is after conditioning '1': Data source is before conditioning resetsignal : SW_Reset
1	CAPTURE_EN	rw	ro	0x0	Enable capturing data in the shift registers (and from ShReg to FW) resetsignal : SW_Reset
0	SW_RST	rw	ro	0x0	Active high soft reset rtl.hw_wp : true

1.15.2 RNG_ETS_CLK_CTRL

0x000D8804

Clock Control register for Entropy Source
rtl.reg_enb : false
display_name : RNG_ETS Clock Configuration

bits	name	s/w	h/w	default	description
26:24	CLK_CONFIG_PNOISE8	rw	ro	0x0	Clock config for phase noise, lane #8 resetsignal : SW_Reset
23:21	CLK_CONFIG_PNOISE7	rw	ro	0x0	Clock config for phase noise, lane #7 resetsignal : SW_Reset
20:18	CLK_CONFIG_PNOISE6	rw	ro	0x0	Clock config for phase noise, lane #6 resetsignal : SW_Reset
17:15	CLK_CONFIG_PNOISE5	rw	ro	0x0	Clock config for phase noise, lane #5 resetsignal : SW_Reset
14:12	CLK_CONFIG_PNOISE4	rw	ro	0x0	Clock config for phase noise, lane #4 resetsignal : SW_Reset
11:9	CLK_CONFIG_PNOISE3	rw	ro	0x0	Clock config for phase noise, lane #3 resetsignal : SW_Reset
8:6	CLK_CONFIG_PNOISE2	rw	ro	0x0	Clock config for phase noise, lane #2 resetsignal : SW_Reset
5:3	CLK_CONFIG_PNOISE1	rw	ro	0x0	Clock config for phase noise, lane #1 resetsignal : SW_Reset
2:0	CLK_CONFIG_PNOISE0	rw	ro	0x0	Clock config for phase noise, lane #0 resetsignal : SW_Reset

1.15.3 RNG_DOWN_SAMPLE_RATIO

0x000D8808

Down Sample Ratio for Entropy Source data
rtl.reg_enb : false
display_name : Down Sample Ratio

bits	name	s/w	h/w	default	description
31:0	ES_DSR	rw	ro	0x0	Down sampling rate for the entropy source output resetsignal : SW_Reset

1.15.4 RNG_ETS_STATUS_STICKY

0x000D880C

Status register for data ready
rtl.reg_enb : false
display_name : Status sticky register

bits	name	s/w	h/w	default	description
28:0	STATUS_STICKY	r/w1c	rw	0x0	Pulse expander when 32 bits are ready. FW should write 1 to each bit to clear it. Bit mapping: 28:20 pnoise_data_b4_xor,

bits	name	s/w	h/w	default	description
					19:17 thermal_data_b4_xor1, 16:14 thermal_data_b4_xor0, 13 pnoise_xor, 12 thermal_xor, 11:0 data_raw resetsignal : SW_Reset

1.15.5 RNG_ETS_LANE_DATA_0

0x000D8810

Lane 0 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 0 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_0	ro	wo	0x0	Raw data from lane 0 of the ETS resetsignal : SW_Reset

1.15.6 RNG_ETS_LANE_DATA_1

0x000D8814

Lane 1 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 1 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_1	ro	wo	0x0	Raw data from lane 1 of the ETS resetsignal : SW_Reset

1.15.7 RNG_ETS_LANE_DATA_2

0x000D8818

Lane 2 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 2 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_2	ro	wo	0x0	Raw data from lane 2 of the ETS resetsignal : SW_Reset

1.15.8 RNG_ETS_LANE_DATA_3

0x000D881C

Lane 3 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 3 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_3	ro	wo	0x0	Raw data from lane 3 of the ETS resetsignal : SW_Reset

1.15.9 RNG_ETS_LANE_DATA_4

0x000D8820

Lane 4 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 4 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_4	ro	wo	0x0	Raw data from lane 4 of the ETS resetsignal : SW_Reset

1.15.10 RNG_ETS_LANE_DATA_5

0x000D8824

Lane 5 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 5 DATA register

bits	name	s/w	h/w	default	description

31:0	LANE_DATA_5	ro	wo	0x0	Raw data from lane 5 of the ETS resetsignal : SW_Reset

1.15.11 RNG_ETS_LANE_DATA_6

0x000D8828

Lane 6 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 6 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_6	ro	wo	0x0	Raw data from lane 6 of the ETS resetsignal : SW_Reset

1.15.12 RNG_ETS_LANE_DATA_7

0x000D882C

Lane 7 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 7 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_7	ro	wo	0x0	Raw data from lane 7 of the ETS resetsignal : SW_Reset

1.15.13 RNG_ETS_LANE_DATA_8

0x000D8830

Lane 8 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 8 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_8	ro	wo	0x0	Raw data from lane 8 of the ETS resetsignal : SW_Reset

1.15.14 RNG_ETS_LANE_DATA_9

0x000D8834

Lane 9 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 9 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_9	ro	wo	0x0	Raw data from lane 9 of the ETS resetsignal : SW_Reset

1.15.15 RNG_ETS_LANE_DATA_10

0x000D8838

Lane 10 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 10 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_10	ro	wo	0x0	Raw data from lane 10 of the ETS resetsignal : SW_Reset

1.15.16 RNG_ETS_LANE_DATA_11

0x000D883C

Lane 11 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 11 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_11	ro	wo	0x0	Raw data from lane 11 of the ETS resetsignal : SW_Reset

1.15.17 RNG_ETS_PH_XOR_DATA

Reg. 0x000D8840

Lane 12 (Pnoise XOR data). (LSB is first data, MSB is last data)
rtl.reg_enb : false
display_name : PHASE XOR DATA register

bits	name	s/w	h/w	default	description
31:0	PH_XOR_DATA	ro	wo	0x0	Data from phase-XOR of the ETS (LSB is first data, MSB is last data). resetsignal : SW_Reset

1.15.18 RNG_ETS_TH_XOR_DATA

Reg. 0x000D8844

Lane 13 (Thermal XOR data). (LSB is first data, MSB is last data)
rtl.reg_enb : false
display_name : THERMAL XOR DATA register

bits	name	s/w	h/w	default	description
31:0	TH_XOR_DATA	ro	wo	0x0	Data from thermal-XOR of the ETS (LSB is first data, MSB is last data). resetsignal : SW_Reset

1.15.19 RNG_ETS_LANE_DATA_14

Reg. 0x000D8848

Lane 14 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 14 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_14	ro	wo	0x0	Raw data from lane 14 of the ETS resetsignal : SW_Reset

1.15.20 RNG_ETS_LANE_DATA_15

Reg. 0x000D884C

Lane 15 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 15 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_15	ro	wo	0x0	Raw data from lane 15 of the ETS resetsignal : SW_Reset

1.15.21 RNG_ETS_LANE_DATA_16

Reg. 0x000D8850

Lane 16 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 16 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_16	ro	wo	0x0	Raw data from lane 16 of the ETS resetsignal : SW_Reset

1.15.22 RNG_ETS_LANE_DATA_17

Reg. 0x000D8854

Lane 17 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 17 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_17	ro	wo	0x0	Raw data from lane 17 of the ETS resetsignal : SW_Reset

1.15.23 RNG_ETS_LANE_DATA_18

Reg. 0x000D8858

Lane 18 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 18 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_18	ro	wo	0x0	Raw data from lane 18 of the ETS resetsignal : SW_Reset

1.15.24 RNG_ETS_LANE_DATA_19

Reg. 0x000D885C

Lane 19 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 19 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_19	ro	wo	0x0	Raw data from lane 19 of the ETS resetsignal : SW_Reset

1.15.25 RNG_ETS_LANE_DATA_20

Reg. 0x000D8860

Lane 20 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 20 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_20	ro	wo	0x0	Raw data from lane 20 of the ETS resetsignal : SW_Reset

1.15.26 RNG_ETS_LANE_DATA_21

Reg. 0x000D8864

Lane 21 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 21 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_21	ro	wo	0x0	Raw data from lane 21 of the ETS resetsignal : SW_Reset

1.15.27 RNG_ETS_LANE_DATA_22

Reg. 0x000D8868

Lane 22 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 22 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_22	ro	wo	0x0	Raw data from lane 22 of the ETS resetsignal : SW_Reset

1.15.28 RNG_ETS_LANE_DATA_23

Reg. 0x000D886C

Lane 23 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 23 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_23	ro	wo	0x0	Raw data from lane 23 of the ETS resetsignal : SW_Reset

1.15.29 RNG_ETS_LANE_DATA_24

Reg. 0x000D8870

Lane 24 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 24 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_24	ro	wo	0x0	Raw data from lane 24 of the ETS resetsignal : SW_Reset

1.15.30 RNG_ETS_LANE_DATA_25

Reg. 0x000D8874

Lane 25 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 25 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_25	ro	wo	0x0	Raw data from lane 25 of the ETS resetsignal : SW_Reset

1.15.31 RNG_ETS_LANE_DATA_26

Reg. 0x000D8878

Lane 26 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 26 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_26	ro	wo	0x0	Raw data from lane 26 of the ETS resetsignal : SW_Reset

1.15.32 RNG_ETS_LANE_DATA_27

Reg. 0x000D887C

Lane 27 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 27 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_27	ro	wo	0x0	Raw data from lane 27 of the ETS resetsignal : SW_Reset

1.15.33 RNG_ETS_LANE_DATA_28

Reg. 0x000D8880

Lane 28 data (LSB is first data, MSB is last data).
rtl.reg_enb : false
display_name : LANE 28 DATA register

bits	name	s/w	h/w	default	description
31:0	LANE_DATA_28	ro	wo	0x0	Raw data from lane 28 of the ETS resetsignal : SW_Reset

1.15.34 RNG_ETS_POWER_VALID

Reg. 0x000D8884

Power valid from the entropy-source
rtl.reg_enb : false
display_name : Power valid register

bits	name	s/w	h/w	default	description
11:0	PWR_VALID	ro	wo	0x0	Power valid from the analog core (per lane). Status only by polling resetsignal : SW_Reset

1.15.35 RNG_RCT_PH_THRSHOLD_0

Reg. 0x000D8888

RCT Threshold for Thermal noise lane 0

rtl.reg_enb : false
display_name : Phase noise RCT threshold 0

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 0 resetsignal : SW_Reset

1.15.36 RNG_RCT_PH_THRSHOLD_1

Reg. 0x000D888C

RCT Threshold for Thermal noise lane 1
rtl.reg_enb : false
display_name : Phase noise RCT threshold 1

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 1 resetsignal : SW_Reset

1.15.37 RNG_RCT_PH_THRSHOLD_2

Reg. 0x000D8890

RCT Threshold for Thermal noise lane 2
rtl.reg_enb : false
display_name : Phase noise RCT threshold 2

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 2 resetsignal : SW_Reset

1.15.38 RNG_RCT_PH_THRSHOLD_3

Reg. 0x000D8894

RCT Threshold for Thermal noise lane 3
rtl.reg_enb : false
display_name : Phase noise RCT threshold 3

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 3 resetsignal : SW_Reset

1.15.39 RNG_RCT_PH_THRSHOLD_4

Reg. 0x000D8898

RCT Threshold for Thermal noise lane 4
rtl.reg_enb : false
display_name : Phase noise RCT threshold 4

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 4 resetsignal : SW_Reset

1.15.40 RNG_RCT_PH_THRSHOLD_5

Reg. 0x000D889C

RCT Threshold for Thermal noise lane 5
rtl.reg_enb : false
display_name : Phase noise RCT threshold 5

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane resetsignal : SW_Reset

1.15.41 RNG_RCT_PH_THRSHOLD_6

0x000D88A0

RCT Threshold for Thermal noise lane 6
rtl.reg_enb : false
display_name : Phase noise RCT threshold 6

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 6 resetsignal : SW_Reset

1.15.42 RNG_RCT_PH_THRSHOLD_7

0x000D88A4

RCT Threshold for Thermal noise lane 7
rtl.reg_enb : false
display_name : Phase noise RCT threshold 7

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 7 resetsignal : SW_Reset

1.15.43 RNG_RCT_PH_THRSHOLD_8

0x000D88A8

RCT Threshold for Thermal noise lane 8
rtl.reg_enb : false
display_name : Phase noise RCT threshold 8

bits	name	s/w	h/w	default	description
15:0	RCT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for phase-noise lane 8 resetsignal : SW_Reset

1.15.44 RNG_APT_PH_THRSHOLD_0

0x000D88AC

Threshold for APT of phase noise lane 0
rtl.reg_enb : false
display_name : Phase noise APT threshold 0

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	APT Threshold for Thermal noise lane 0 resetsignal : SW_Reset

1.15.45 RNG_APT_PH_THRSHOLD_1

0x000D88B0

APT Threshold for Thermal noise lane 1
rtl.reg_enb : false
display_name : Phase noise APT threshold 1

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 1 resetsignal : SW_Reset

1.15.46 RNG_APT_PH_THRSHOLD_2

0x000D88B4

APT Threshold for Thermal noise lane 2
rtl.reg_enb : false
display_name : Phase noise APT threshold 2

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 2 resetsignal : SW_Reset

1.15.47 RNG_APT_PH_THRSHOLD_3

Reg. 0x000D88B8

APT Threshold for Thermal noise lane 3
rtl.reg_enb : false
display_name : Phase noise APT threshold 3

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 3 resetsignal : SW_Reset

1.15.48 RNG_APT_PH_THRSHOLD_4

Reg. 0x000D88BC

APT Threshold for Thermal noise lane 4
rtl.reg_enb : false
display_name : Phase noise APT threshold 4

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 4 resetsignal : SW_Reset

1.15.49 RNG_APT_PH_THRSHOLD_5

Reg. 0x000D88C0

APT Threshold for Thermal noise lane 5
rtl.reg_enb : false
display_name : Phase noise APT threshold 5

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane resetsignal : SW_Reset

1.15.50 RNG_APT_PH_THRSHOLD_6

Reg. 0x000D88C4

APT Threshold for Thermal noise lane 6
rtl.reg_enb : false
display_name : Phase noise APT threshold 6

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 6 resetsignal : SW_Reset

1.15.51 RNG_APT_PH_THRSHOLD_7

Reg. 0x000D88C8

APT Threshold for Thermal noise lane 7
rtl.reg_enb : false
display_name : Phase noise APT threshold 7

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 7 resetsignal : SW_Reset

1.15.52 RNG_APT_PH_THRSHOLD_8

Reg. 0x000D88CC

APT Threshold for Thermal noise lane 8
rtl.reg_enb : false
display_name : Phase noise APT threshold 8

bits	name	s/w	h/w	default	description

bits	name	s/w	h/w	default	description
9:0	APT_PH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for phase-noise lane 8 resetsignal : SW_Reset

1.15.53 RNG_RCT_TH_THRSHOLD_0

Reg. 0x000D88D0

RCT Threshold for Thermal noise lane 0
rtl.reg_enb : false
display_name : Thermal noise RCT threshold 0

bits	name	s/w	h/w	default	description
10:0	RCT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for Thermal-noise lane 0 resetsignal : SW_Reset

1.15.54 RNG_RCT_TH_THRSHOLD_1

Reg. 0x000D88D4

RCT Threshold for Thermal noise lane 1
rtl.reg_enb : false
display_name : Thermal noise RCT threshold 1

bits	name	s/w	h/w	default	description
10:0	RCT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for Thermal-noise lane 1 resetsignal : SW_Reset

1.15.55 RNG_RCT_TH_THRSHOLD_2

Reg. 0x000D88D8

RCT Threshold for Thermal noise lane 2
rtl.reg_enb : false
display_name : Thermal noise RCT threshold 2

bits	name	s/w	h/w	default	description
10:0	RCT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for Thermal-noise lane 2 resetsignal : SW_Reset

1.15.56 RNG_APT_TH_THRSHOLD_0

Reg. 0x000D88DC

APT Threshold for Thermal noise lane 0
rtl.reg_enb : false
display_name : Thermal noise APT threshold 0

bits	name	s/w	h/w	default	description
9:0	APT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for Thermal-noise lane 0 resetsignal : SW_Reset

1.15.57 RNG_APT_TH_THRSHOLD_1

Reg. 0x000D88E0

APT Threshold for Thermal noise lane 1
rtl.reg_enb : false
display_name : Thermal noise APT threshold 1

bits	name	s/w	h/w	default	description
9:0	APT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for Thermal-noise lane 1 resetsignal : SW_Reset

1.15.58 RNG_APT_TH_THRSHOLD_2

Reg. 0x000D88E4

APT Threshold for Thermal noise lane 2
rtl.reg_enb : false

display_name : Thermal noise APT threshold 2

bits	name	s/w	h/w	default	description
9:0	APT_TH_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for Thermal-noise lane 2 resetsignal : SW_Reset

1.15.59 RNG_RCT_XOR_THRSHOLD

Reg. 0x000D88E8

RCT Threshold for XOR lane
rtl.reg_enb : false
display_name : XOR RCT threshold

bits	name	s/w	h/w	default	description
10:0	RCT_XOR_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: RCT test for XOR resetsignal : SW_Reset

1.15.60 RNG_APT_XOR_THRSHOLD

Reg. 0x000D88EC

APT Threshold for XOR lane
rtl.reg_enb : false
display_name : XOR APT threshold

bits	name	s/w	h/w	default	description
9:0	APT_XOR_THRESHOLD	rw	ro	0x0	Threshold for Entropy-Source Health-Test: APT test for XOR resetsignal : SW_Reset

1.15.61 RNG_ERR_STATUS

Reg. 0x000D88F0

General error status register with status bit for each error event
rtl.reg_enb : false
display_name : RNG Error Status

bits	name	s/w	h/w	default	description
0	DATA_LOST	r/w1c	rw	0x0	Indicates when data is lost - data arrived but the previous data wasn't read yet. resetsignal : SW_Reset we : true

1.15.62 RNG_FATAL_STATUS

Reg. 0x000D88F4

Health-tests failure status register with a status bit for each failure. Failures described in NIST 800-90B
rtl.reg_enb : false
display_name : RNG Fatal Status

bits	name	s/w	h/w	default	description
25	ETS_XOR_APT_FATAL	ro	wo	0x0	After-XOR noise has an APT health test failure. resetsignal : SW_Reset we : true
24	ETS_XOR_RCT_FATAL	ro	wo	0x0	After-XOR has an RCT health test failure. resetsignal : SW_Reset we : true
23:15	ETS_PH_APT_FATAL	ro	wo	0x0	Phase noise has an APT health test failure. resetsignal : SW_Reset we : true
14:6	ETS_PH_RCT_FATAL	ro	wo	0x0	Phase noise has an RCT health test failure. resetsignal : SW_Reset we : true
5:3	ETS_TH_APT_FATAL	ro	wo	0x0	Thermal noise has an APT health test failure. resetsignal : SW_Reset we : true
2:0	ETS_TH_RCT_FATAL	ro	wo	0x0	Thermal noise has an RCT health test failure. resetsignal : SW_Reset we : true

1.15.63 RNG_INTR_STATE

0x000D88F8

General purpose interrupt status register with each bit corresponding to an interrupt port. There is 1 bit for consolidated alert and error events each, and a dedicated bit for any other status events that need to be routed as interrupts. All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be re-latched.

rtl.reg_enb : false
display_name : RNG Interrupt State

bits	name	s/w	h/w	default	description
1	ERR	r/w1c	rw	0x0	Error interrupt. Status indicating an error has been detected. This bit is a consolidation (OR) of all error events in RNG_ERR_STATUS. resetsignal : SW_Reset
0	FATAL	r/w1c	rw	0x0	Fatal interrupt. A health-test failure has been detected. This bit is a consolidation (OR) of all fatal events in RNG_FATAL_STATUS resetsignal : SW_Reset

1.15.64 RNG_INTR_ENABLE

0x000D88FC

General purpose interrupt enable status register with a bit to mask/unmask each interrupt port.

rtl.reg_enb : false
display_name : RNG Interrupt Enable
'intr.mask' : Identifies the 'RNG_INTR_STATE' for the interrupt logic.

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x1	Error interrupt enable 0: do not generate interrupt 1: generate interrupt if RNG_INT_STATE.ERR resetsignal : SW_Reset
0	FATAL	rw	ro	0x1	Fatal interrupt enable 0: do not generate interrupt 1: generate interrupt if RNG_INT_STATE.FATAL resetsignal : SW_Reset

1.15.65 RNG_INTR_TEST

0x000D8900

General purpose interrupt test register with a bit to force each interrupt for test and debug.

rtl.reg_enb : false
dontcompare : true
display_name : RNG Interrupt Test

bits	name	s/w	h/w	default	description
1	ERR	rw	ro	0x0	Assert error interrupt 0: do not force interrupt 1: force RNG_INTR_STATE.ERR resetsignal : SW_Reset
0	FATAL	rw	ro	0x0	Assert fatal interrupt 0: do not force interrupt 1: force RNG_INTR_STATE.FATAL resetsignal : SW_Reset

1.15.66 RNG_STATUS

0x000D8904

General purpose status register

rtl.reg_enb : false
display_name : RNG Status

bits	name	s/w	h/w	default	description
0	SW_RST_DONE	ro	wo	0x1	Software reset completion status 0: Reset not successful 1: Reset successful rtl.hw_set : true rtl.hw_clear : true we : true

End RegGroup

1.16 ROT_DMA

RegGrp

0x000E0000 - 0x000E5067

decode_size : 0x00060000
chapter : 1.9, Security Subsystem, ROT DMA 0
blockgroup : SECUREPROCESSOR
revision : revision: 1502e5d

1.16.1 DMA

RegGrp

0x000E0000 - 0x000E5067

'stride' : '4096' Specifies the address stride when instantiating an array of components
'Count' : 'DMA' will repeat '6' times.

count	0	1	2	3	4	5
address	0xE0000	0xE1000	0xE2000	0xE3000	0xE4000	0xE5000

1.16.1.1 PARAMS

Reg.

0x000E0000

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
8:0	FIFO_BYTE_DEPTH	ro	wo	0x20	Byte depth of the DMA FIFO. Possible values are 32, 64, 256 If AVAIL_LEVEL_SRC/DST are set above FIFO_BYTE_DEPTH or to zero, hardware resets the AVAIL_LEVEL_SRC/DST 16 Reset values: • DMA[0-1].FIFO_BYTE_DEPTH = 64 for Optimus and CSSD • DMA[0-1].FIFO_BYTE_DEPTH = 256 for HDD • DMA[2-5].FIFO_BYTE_DEPTH = 32 for all SoCs undef_mask : 511 resetsignal : SW_Reset

1.16.1.2 CFG

Reg.

0x000E0004

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
17:9	AVAIL_LEVEL_SRC	rw	rw	0x10	Hardware uses this value to help calculate its default read transfer word count Restrictions on AVAIL_LEVEL_SRC programming: • AVAIL_LEVEL_SRC must a multiple of BYTES_PER_BEAT • AVAIL_LEVEL_SRC must be less than or equal to DMA_PARAMS.FIFO_BYTE_DEPTH * BYTES_PER_BEAT Hardware repairs an invalid setting by using 0x10

					when the invalid setting is detected hardware corrects the field in place Otherwise, hardware waits until the DMA FIFO has AVAIL_LEVEL_SRC bytes of space available before transferring from the source address to the DMA FIFO Changing the default value of AVAIL_LEVEL_SRC is desirable in many cases, espcially when forcing DMA to do large bursts from the SRC address. However, firmware must understand the section in the DMA specification related to DMA Deadlock. Incorrect settings on AVAIL_LEVEL_SRC/DST cause a Deadlock. Additionally in FIFO SRC Mode, AVAIL_LEVEL_SRC must be less than or equal to the size of the SRC_FIFO. If it set to a value greater than the SRC_FIFO byte depth, the DMA will hang. For example, the AES FIFO is 0x20 bytes deep. A AVAIL_LEVEL_SRC setting greater than 0x20 will cause the DMA to hang. dontcompare : true resetsignal : SW_Reset
8:0	AVAIL_LEVEL_DST	rw	rw	0x10	Hardware uses this value to help calculate its default write transfer byte count Restrictions on AVAIL_LEVEL_DST programming: • AVAIL_LEVEL_DST must a multiple of BYTES_PER_BEAT • AVAIL_LEVEL_DST must be less than or equal to DMA_PARAMS.FIFO_BYTE_DEPTH * BYTES_PER_BEAT Hardware repairs an invalid setting by writing the value, 0x10. when the invalid setting is detected hardware corrects the field in place Otherwise, hardware waits until the DMA FIFO has AVAIL_LEVEL_DST bytes of space available before transferring from the DMA FIFO to the destination address Changing the default value of AVAIL_LEVEL_DST is desirable in many cases, espcially when forcing DMA to do large bursts to the DST address. However, firmware must understand the section in the DMA specification related to DMA Deadlock. Incorrect settings on AVAIL_LEVEL_SRC/DST cause a Deadlock. Additionally in FIFO DST Mode, AVAIL_LEVEL_DST must be less than or equal to the size of the DST_FIFO. If it set to a value greater than the DST_FIFO byte depth, the DMA will hang. For example, the AES FIFO is 0x20 bytes deep. A AVAIL_LEVEL_DST setting greater than 0x20 will cause the DMA to hang. dontcompare : true resetsignal : SW_Reset

1.16.1.3 CTRL

0x000E0008

DMA Control register resets DMA engine and sets the CMD Level for command AVAIL notification

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
5:4	CMD_LEVEL	rw	ro	0x1	Hardware drives DMA_STATUS.CMD_AVAIL_STATUS to 1 when CMD_AVAIL_COUNT greater or equal to DMA_CFG.CMD_LEVEL. CMD_LEVEL = 2 -> CMD FIFO Empty CMD_LEVEL = 1 -> CMD FIFO Not Full CMD_LEVEL = 0 -> Will always interrupt

					dontcompare : true resetsignal : SW_Reset
1	ABORT	rw	ro	0x0	Before resetting the DMA, regardless of using SW_RST or system reset, firmware should assert ABORT until STATUS.QUIESCE == 1. Clear the abort before resetting resetsignal : SW_Reset
0	SW_RST	rw	ro	0x0	Active HIGH soft reset - safe reset of DMA state when used in conjunction with ABORT/QUIESCE. Firmware should 1. Assert ABORT 2. Wait for DMA_STATUS.QUIESCE 3. Set SW_RST SW_RST affects • all DMA registers (including SW_RST) • DMA state machines. • DMA Command FIFO • DMA Completion FIFO • DMA Data Path Barrel Shifter SW_RST does not touch the DMA bus masters dontcompare : true resetsignal : SW_Reset

1.16.1.4 SRC

Reg. 0x000E000C

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	rw	ro	0x0	DMA source address. In FIFO Modes, ADDR specifies the fixed SRC FIFO address. In all other modes, ADDR specifies the starting byte address of the source data. resetsignal : SW_Reset

1.16.1.5 DST

Reg. 0x000E0010

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	rw	ro	0x0	DMA destination address. In FIFO Modes, ADDR specifies the fixed DST FIFO address. In all other modes, ADDR specifies the starting byte address of the destination data. resetsignal : SW_Reset

1.16.1.6 LEN

Reg. 0x000E0014

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
25:0	BYTE_LEN	rw	ro	0x0	Byte length of the primary data transfer Maximum: 64MB - 32 bytes resetsignal : SW_Reset

1.16.1.7 MODE

Reg. 0x000E0018

Transfer type. When this register is written, the DMA Command FIFO is pushed.
rtl.reg_enb : true
no_reg_bit_bash_test : true

<table>
<tr><th>bits</th><th>name</th><th>s/w</th><th>h/w</th><th>default</th><th>description</th></tr>
<tr><td>25:20</td><td>JOB_ID</td><td>rw</td><td>ro</td><td>0x0</td><td>If CMD_COMPLETE flag is set, this value will be
written to the Completion FIFO
The value can also be read at the FIFO_TOP for debug purposes.
resetsignal : SW_Reset</td></tr>
<tr><td>18:16</td><td>FIFO_SEL_DST</td><td>rw</td><td>ro</td><td>0x0</td><td>When the DST_TYPE is DST_FIFO_32 or DST_FIFO_8 this field
selects the desired FIFO DMA Flow control signals from the desired DST FIFO. A DST FIFO is also known as a CRYPTO IN FIFO or a TX FIFO.
The DST FIFO address must be specified in DMA.DST.ADDR.
When the DST_TYPE does not indicate a FIFO mode, this field is ignored

enum:FIFO_FLOW_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>FIFO_FLOW_SHA</td><td>0</td><td>DMA Flow Control from SHA FIFO</td></tr>
<tr><td>FIFO_FLOW_AES</td><td>1</td><td>DMA Flow Control from AES FIFO</td></tr>
<tr><td>FIFO_FLOW_GCE</td><td>2</td><td>DMA Flow Control from AES GCM FIFO</td></tr>
<tr><td>FIFO_FLOW_I2C</td><td>3</td><td>DMA FLow Control from TX/RX I2C FIFO</td></tr>
<tr><td>FIFO_FLOW_UART</td><td>4</td><td>DMA Flow Control from TX/RX UART FIFO</td></tr>
<tr><td>FIFO_FLOW_SPI</td><td>5</td><td>DMA FLow Control from TX/RX SPI FIFO</td></tr>
</table>
encode : FIFO_FLOW_e
resetsignal : SW_Reset</td></tr>
<tr><td>14:12</td><td>FIFO_SEL_SRC</td><td>rw</td><td>ro</td><td>0x0</td><td>When the SRC_TYPE is SRC_FIFO_32 or SRC_FIFO_8 this field
selects the desired FIFO DMA Flow control signals from the desired SRC FIFO. A SRC FIFO is also known as a CRYPTO OUT FIFO or an RX FIFO.
The SRC FIFO address must be specified in DMA.SRC.ADDR.
When the SRC_TYPE does not indicate a FIFO mode, this field is ignored

enum:FIFO_FLOW_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>FIFO_FLOW_SHA</td><td>0</td><td>DMA Flow Control from SHA FIFO</td></tr>
<tr><td>FIFO_FLOW_AES</td><td>1</td><td>DMA Flow Control from AES FIFO</td></tr>
<tr><td>FIFO_FLOW_GCE</td><td>2</td><td>DMA Flow Control from AES GCM FIFO</td></tr>
<tr><td>FIFO_FLOW_I2C</td><td>3</td><td>DMA FLow Control from TX/RX I2C FIFO</td></tr>
</table></td></tr>
</table>

<table>
<tr><th></th><th></th><th></th><th></th><th></th><th></th></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td><table><tr><td>FIFO_FLOW_UART</td><td>4</td><td>DMA Flow Control from TX/RX UART FIFO</td></tr><tr><td>FIFO_FLOW_SPI</td><td>5</td><td>DMA FLow Control from TX/RX SPI FIFO</td></tr></table>encode : FIFO_FLOW_e
resetsignal : SW_Reset</td></tr>
<tr><td>8</td><td>WAIT_WRITE_RESP</td><td>rw</td><td>ro</td><td>0x1</td><td>1: The command will not be popped from the command FIFO until all
write responses have been received from the bus master. Setting this bit to 1 guarantees that data have made it to memory before
finishing the DMA.
0: DMA completes once all teh writes are posted but not necessily completed.
Chained DMAs that have mutually exclusive source and destination addresses
can benefit in performance by not waiting for write response
resetsignal : SW_Reset</td></tr>
<tr><td>7</td><td>NATIVE_AXI</td><td>rw</td><td>ro</td><td>0x0</td><td>1: optimal performance - awsize will always be the maximum.
For example, for a 128-bit AXI bus, awsize == 4. This setting
allows for all partial writes (at the head or tail of a transfer) to be done in a single clock.
0: prevent overwriting of data for partial accesses where the written byte_count is less than the full bus width (partial writes).
The master breaks a write request that is less the the number of
beats into multiple bus transactions each with
• awsize matches the bytes to be transferred (i.e. awsize = 2 -> 4 bytes)
• 2 byte transfers must correspond to awaddr[0] = 0
• 4 byte transfers must correspond to awaddr[1:0] = 0
• 8 byte transfers must correspond to awaddr[2:0] = 0
• wstrb will exactly correspond to awaddr and awsize
• awlen = 0 for all sub transfers
Only used for DMA 0 and DMA 1 AXI transfers when the destination address is
outside the RoT (e.g. destination address >= 0x10000000).
When the external AXI bus connects to a slave that is not native AXI-4, setting NATIVE_AXI = 0
may prevent inadvertent overwriting of data. For example, some AXI to AHB translators, like the NIC400, require that the transfer bytes transferred are always powers of 2 and that the address
alignent and wstrb exactly match the aw_size.
Regardless of the state of this field, DMA access to SCRATCH Memory always operate as if NATIVE_AXI = 1. All other RoT
peripheral accesses operate as if NATIVE_AXI = 0 to accomodate the known
NIC AXI to AHB translation within RoT.
Note that read transfers will always read full beats, even if only a single byte
is required in the beat.
resetsignal : SW_Reset</td></tr>
<tr><td>6</td><td>FW_FLOW_CONTROL</td><td>rw</td><td>ro</td><td>0x0</td><td>When the DMA Controller works on a command with FW_FLOW_CONTROL, set, it will not start working
on the command until its JOB ID is written (along with VALID) into</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>the FW_FLOW_HANDSHAKE.JOB_ID/VALID fields. The DMA_STATUS.FW_FLOW_WAIT will
be set until the command can be processed.
resetsignal : SW_Reset</td></tr>
<tr><td>5</td><td>CMD_COMPLETE</td><td>rw</td><td>ro</td><td>0x0</td><td>When CMD_COMPLETE is set, the JOB_ID is written to the CMD_COMPLETION FIFO when the command completes.
If this flag is not set, the command will complete without notice to firmware.
resetsignal : SW_Reset</td></tr>
<tr><td>4:3</td><td>SRC_TYPE</td><td>rw</td><td>ro</td><td>0x0</td><td>Indicates whether source transfer is FIFO mode or Memory Mode
If FIFO Mode SRC is AES_DATA_OUT
enum:SRC_TYPE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>SRC_MEM</td><td>0</td><td>Memory - No Flow Control Bus width maximized</td></tr>
<tr><td>SRC_FIFO_8</td><td>1</td><td>FIFO Byte Mode - Peripheral RX Flow Control. SRC data comes from an 8-bit RX FIFO</td></tr>
<tr><td>SRC_FIFO_32</td><td>2</td><td>AES Output FIFO Flow Control, FIFO Mode transfers - SRC = AES_DATA_OUT</td></tr>
<tr><td>SRC_FILL</td><td>3</td><td>FILL Mode - SRC data comes from DMA_FILL_VALUE register (DMA_FILL_VALUE is not part of DMA CMD)</td></tr>
</table>
encode : SRC_TYPE_e
resetsignal : SW_Reset</td></tr>
<tr><td>2:0</td><td>DST_TYPE</td><td>rw</td><td>ro</td><td>0x0</td><td>Indicates whether source transfer is FIFO mode or Memory Mode
If FIFO Mode - selects approriate hardware flow control source: AES or SHA
• Selects DST address either AES_DATA_IN or SHA_MSG
• Selects DST SHA_GEN_DIGEST or SHA_GEN_CONTEXT for final FIFO transfer
enum:DST_TYPE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DST_MEM</td><td>0</td><td>Memory - No Flow Control Bus width maximized</td></tr>
<tr><td>DST_FIFO_8</td><td>1</td><td>FIFO Byte Mode - Peripheral TX flow control. DST data goes to an 8-bit TX FIF</td></tr>
</table>
</td></tr>
</table>

		O
DST_FIFO_32	2	Input FIFO Flow Control, FIFO Mode
DST_FIFO_SHA_DIGEST	3	SHA Input FIFO Flow Control, FIFO Mode - DST = SHA_MSG/SHA_GEN_DIGEST
DST_FIFO_SHA_CONTEXT	4	SHA Input FIFO Flow Control, FIFO Mode - DST = SHA_MSG/SHA_GEN_CONTEXT
DST_FILL_VERIFY	5	No Destination. Data are compared with DMA_FILL_VALUE. BYTE LEN must be multiple of 4

encode : DST_TYPE_e
resetsignal : SW_Reset

1.16.1.8 FW_FLOW_HANDSHAKE

Reg. 0x000E001C

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
8	VALID	rw	rw	0x0	Firmware sets VALID flag when it writes the JOB_ID field. Hardware clears the VALID bit once it has start the DMA command. rtl.hw_clear : true resetsignal : SW_Reset
5:0	JOB_ID	rw	ro	0x0	Firmware writes the JOBID of the command that is allowed to start. Hardware will compare against its JOB_ID when the VALID bit is set. resetsignal : SW_Reset

1.16.1.9 COMPLETION_FIFO

Reg. 0x000E0020

DMA Complete FIFO - read JOB_ID at the FIFO output and number of valid entries in the Completion FIFO. Write to pop the FIFO. Hardware ignores writes when FIFO is empty
rtl.reg_enb : false
'COMPLETION_FIFO' is an external.

bits	name	s/w	h/w	default	description
9:8	COMPLETION_COUNT	rw	rw	0x0	number of entries in the Completion FIFO (0,1 or 2) dontcompare : true resetsignal : SW_Reset
5:0	JOB_ID	rw	rw	0x0	JOB_ID from Command FIFO is written to Completion FIFO when CMD_COMPLETE flag is set Write to this register pops the FIFO dontcompare : true resetsignal : SW_Reset

1.16.1.10 FILL_VALUE

Reg. 0x000E0024

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	VALUE	rw	ro	0x0	When DMA_MODE.SRC_TYPE = SRC_FILL: destination memory is filled with VALUE When DMA_MODE.DST_TYPE = DST_FILL_VERIFY: source_data AND ~MASK is compared with VALUE AND ~MASK **Note** DMA_FILL_VALUE is not pushed into the Command FIFO as part of the DMA CMD, so consecutive fill or fill verify commands using *differnt* DMA_FILL_VALUE must either wait till each command completes or use the FW_FLOW_CONTROL before changing DMA_FILL_VALUE per command resetsignal : SW_Reset

1.16.1.11 FILL_FAIL_ADDR

0x000E0028

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	ro	wo	0x0	When a DMA_MODE.DST_FILL_VERIFY operation finds a mismatch, the first failure is captured in ADDR. This field is only valid when DMA_FILL_FAIL_COUNT.COUNT >0 and DMA_INTR_STATE.FILL_FAIL_VERIFY == 1. It is cleared along with DMA_INTR_STATE.FILL_FAIL_VERIFY resetsignal : SW_Reset

1.16.1.12 FILL_FAIL_COUNT

0x000E002C

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
7:0	COUNT	ro	wo	0x0	When DMA_MODE.DST_FILL_VERIFY==1, any mismatch within a data beat increments this counter up to 255. For 128-bit DMA0 or DMA1 any 1-16 detected byte errors in a data beat increments this count by 1. For other 32-bit DMAs 1 to 4 detected byte errors per data beat increment the count by 1. This field is cleared along with DMA_INTR_STATE.FILL_FAIL_VERIFY resetsignal : SW_Reset

1.16.1.13 STATUS

0x000E0030

This register contains byte length.
rtl.reg_enb : false
display_name : General Control Register

bits	name	s/w	h/w	default	description
31	BUSY	ro	wo	0x0	0: DMA is idle. 1: DMA is busy. resetsignal : SW_Reset
30	FW_FLOW_WAIT	ro	wo	0x0	DMA Engine is waiting for firmware to write its JOB_ID and VALID into the FW_FLOW_HANDSHAKE register. resetsignal : SW_Reset
29	CMD_COMPLETE_WAIT	ro	wo	0x0	DMA Engine is waiting for firmware to pop the completion FIFO. It is full. resetsignal : SW_Reset
28	QUIESCE	ro	wo	0x1	DMA Engine detects ABORT, has no outstanding bus requests, and all previous requests are complete. When

					QUIESCE is 1, it is safe to reset the DMA Controller resetsignal : SW_Reset
9	COMPLETION_AVAIL_STATUS	ro	wo	0x0	1: The Completion FIFO is not empty 0: The Completion FIFO is empty; resetsignal : SW_Reset
8	CMD_AVAIL_STATUS	ro	wo	0x1	1: CMD_AVAIL_COUNT greater or equal to DMA_CFG.CMD_LEVEL 0: CMD_AVAIL_COUNT less than DMA_CFG.CMD_LEVEL resetsignal : SW_Reset
5:4	COMPLETION_AVAIL_COUNT	ro	wo	0x0	Number of Completions available to be popped from the Completion FIFO resetsignal : SW_Reset
1:0	CMD_AVAIL_COUNT	ro	wo	0x2	Number of command available to be pushed into Command FIFO resetsignal : SW_Reset

1.16.1.14 ERR_STATUS

Reg. 0x000E0034

This register contains error information.
Some events are fatal errors. When detected, the block ceases operations
and requires a reset before re-use. Other events are non-fatal.
When detected, the status can be
cleared by software by writing 1 to it
rtl.reg_enb : false
display_name : DMA Error Status Register

bits	name	s/w	h/w	default	description
7	MPU_READ_VIOLATION	ro	wo	0x0	An MPU Read Violation was detected. The DMA engine aborts when it detects an MPU Vioaltion. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 (expected immediately) 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) resetsignal : SW_Reset
6	MPU_WRITE_VIOLATION	ro	wo	0x0	An MPU Write Violation was detected. The DMA engine aborts when it detects an MPU Vioaltion. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 (expected immediately) 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) resetsignal : SW_Reset
5	DEADLOCK	ro	wo	0x0	AVAIL_LEVEL_SRC/DST settings have a caused a deadlock condition. To clear status: 1. Set ROT_DMA.CTRL.ABORT = 1 2. Poll for ROT_DMA.STATUS.QUIESCE == 1 3. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) resetsignal : SW_Reset
4	BUS_ERROR_WRITE	ro	wo	0x0	AXI or AHB RESP Error occured during a DMA write transfer. Query BUS_ERROR_WRITE_STATUS for details. The DMA engine aborts when it detects a bus error. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) resetsignal : SW_Reset

3	BUS_ERROR_READ	ro	wo	0x0	AXI or AHB RESP Error occured during a DMA read transfer. Query BUS_ERROR_READ_STATUS for details. The DMA engine aborts when it detects a bus error. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) resetsignal : SW_Reset
2	DATA_FIFO_UNDR	ro	rw	0x0	Data FIFO under run detected. Indicates a fatal hardware error and should never occur. The DMA engine aborts when it detects a Data FIFO Underrun. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) rtl.hw_set : true resetsignal : SW_Reset
1	DATA_FIFO_OVFL	ro	rw	0x0	DATA FIFO overflow detected (fatal)Indicates a fatal hardware error and should never occur. The DMA engine aborts when it detects a Data FIFO Overflow. To clear status: 1. Wait for ROT_DMA.STATUS.QUIESCE == 1 2. Set ROT_DMA.CTRL.SW_RST = 1 This status will clear on soft reset (fatal) rtl.hw_set : true resetsignal : SW_Reset
0	CMD_FIFO_OVFL	r/w1c	rw	0x0	Firmware pushed the Command FIFO when full. The data were not pushed to the FIFO (non-fatal) rtl.hw_set : true resetsignal : SW_Reset

1.16.1.15 BUS_ERROR_READ_STATUS

0x000E0038

Bus error status for reads
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:2	ADDR	ro	wo	0x0	upper 30 bits of address that caused the bus error resetsignal : SW_Reset
1:0	RESP	ro	wo	0x0	resp from bus resetsignal : SW_Reset

1.16.1.16 BUS_ERROR_WRITE_STATUS

0x000E003C

Bus error status for writes
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:2	ADDR	ro	wo	0x0	upper 30 bits of address that caused the bus error resetsignal : SW_Reset
1:0	RESP	ro	wo	0x0	resp from bus resetsignal : SW_Reset

1.16.1.17 INTR_STATE

0x000E0040

DMA interrupt status register with each bit corresponding to an interrupt port;
There is 1 bit for consolidated alert and error events each, and a
dedicated bit for any other status events that need to be routed as

interrupts. All the bits are latched and are cleared by writing 1. If the input event condition still persists, the bit field will be re-latched.
rtl.reg_enb : true
no_reg_bit_bash_test : true

bits	name	s/w	h/w	default	description
4	FILL_VERIFY_FAIL	r/w1c	rw	0x0	FILL Verify detects one or more mismatches. Refer to DMA_FILL_FAIL.ADDR and DMA_FILL_FAIL_COUNT for details. The DMA Engine will not finish its current command until this status is cleared by firmware. In this case, the contents of DMA_CMD_TOP* registers reflect the failed command. Once cleared, the DMA will pop the command FIFO and push completion status if the command so indicated with the cmd_complete bit. rtl.hw_set : true resetsignal : SW_Reset
3	COMPLETION_FIFO	r/w1c	rw	0x0	The Completion FIFO is not empty rtl.hw_set : true resetsignal : SW_Reset
2	CMD_FIFO_AVAIL	r/w1c	rw	0x1	The Command FIFO is available for pushing. Specifically, the number of available entries is greater or equal to the DMA_CONFIG.CMD_LEVEL rtl.hw_set : true resetsignal : SW_Reset
1	ERR	r/w1c	rw	0x0	Error interrupt. Status indicating an error has been detected. This bit is a consolidation (OR) of all error events in DMA_ERR_STATUS. rtl.hw_set : true resetsignal : SW_Reset
0	ALERT	r/w1c	rw	0x0	ALert interrupt. A security relevant error has been detected. This bit is a consolidation (OR) of all alert events in DMA_ALERT_STATUS. rtl.hw_set : true resetsignal : SW_Reset

1.16.1.18 INTR_ENABLE

Reg. 0x000E0044

DMA interrupt enable status register with a bit to enable/disab;e each interrupt port.
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
4	FILL_VERIFY_FAIL	rw	ro	0x0	FILL Verify failure interrupt enable resetsignal : SW_Reset
3	COMPLETION_FIFO	rw	ro	0x0	Completion FIFO interrupt enable resetsignal : SW_Reset
2	CMD_FIFO_AVAIL	rw	ro	0x0	CMD_FIFO_AVAIL interrupt enable dontcompare : true resetsignal : SW_Reset
1	ERR	rw	ro	0x1	Error interrupt enable. 0: do not generate interrupt. 1: generate interrupt if DMA_INTR_STATE.ERR resetsignal : SW_Reset
0	ALERT	rw	ro	0x1	Alert interrupt enable. 0: do not generate interrupt. 1: generate interrupt if DMA_INTR_STATE.ALERT resetsignal : SW_Reset

1.16.1.19 INTR_TEST

Reg. 0x000E0048

General purpose interrupt test register with a bit to force each interrupt for test and debug.
rtl.reg_enb : false
dontcompare : true
display_name : General Interrupt Test Register

bits	name	s/w	h/w	default	description
4	FILL_VERIFY_FAIL	rw	ro	0x0	Assert FILL_VERIFY_FAIL interrupt resetsignal : SW_Reset
3	COMPLETION_FIFO	rw	ro	0x0	Assert COMPLETION_FIFO interrupt. resetsignal : SW_Reset
2	CMD_FIFO_AVAIL	rw	ro	0x0	Assert CMD_FIFO_AVAIL interrupt resetsignal : SW_Reset
1	ERR	rw	ro	0x0	Assert error interrupt. 0: do not force interrupt. 1: force DMA_INT_STATE.ERR resetsignal : SW_Reset
0	ALERT	rw	ro	0x0	Assert alert interrupt. 0: do not force interrupt. 1: force DMA_INT_STATE.ALERT resetsignal : SW_Reset

1.16.1.20 CMD_FIFO_TOP_SRC

0x000E004C

This register shows the top of the command FIFO.Contents are only valid when the Command FIFO is not empty.
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	ro	wo	0x0	Top of Command FIFO - DMA source address resetsignal : SW_Reset

1.16.1.21 CMD_FIFO_TOP_DST

0x000E0050

This register shows the top of the command FIFO.Contents are only valid when the Command FIFO is not empty.
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	ADDR	ro	wo	0x0	Top of Command FIFO - DMA destination address resetsignal : SW_Reset

1.16.1.22 CMD_FIFO_TOP_LEN

0x000E0054

This register shows the top of the command FIFO.Contents are only valid when the Command FIFO is not empty.
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
25:0	BYTE_LEN	ro	wo	0x0	Top of Command FIFO - Byte length of the primary data transfer resetsignal : SW_Reset

1.16.1.23 CMD_FIFO_TOP_MODE

0x000E0058

This register shows the top of the command FIFO.Contents are only valid when the Command FIFO is not empty.
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
25:20	JOB_ID	ro	wo	0x0	Top of Command FIFO - JOB ID resetsignal : SW_Reset
18:16	FIFO_SEL_DST	ro	wo	0x0	Top of Command FIFO enum:FIFO_FLOW_e

Name	Value	Description
FIFO_FLOW_SHA	0	DMA Flow Control from SHA FIFO
FIFO_FLOW_A	1	DMA Flow Co

ES		ntrol from AES FIFO
FIFO_FLOW_GCE	2	DMA Flow Control from AES GCM FIFO
FIFO_FLOW_I2C	3	DMA FLow Control from TX/RX I2C FIFO
FIFO_FLOW_UART	4	DMA Flow Control from TX/RX UART FIFO
FIFO_FLOW_SPI	5	DMA FLow Control from TX/RX SPI FIFO

encode : FIFO_FLOW_e
resetsignal : SW_Reset

14:12	FIFO_SEL_SRC	ro	wo	0x0	Top of Command FIFO

enum:FIFO_FLOW_e

Name	Value	Description
FIFO_FLOW_SHA	0	DMA Flow Control from SHA FIFO
FIFO_FLOW_AES	1	DMA Flow Control from AES FIFO
FIFO_FLOW_GCE	2	DMA Flow Control from AES GCM FIFO
FIFO_FLOW_I2C	3	DMA FLow Control from TX/RX I2C FIFO
FIFO_FLOW_UART	4	DMA Flow Control from TX/RX UART FIFO
FIFO_FLOW_SPI	5	DMA FLow Control from TX/RX SPI FIFO

encode : FIFO_FLOW_e
resetsignal : SW_Reset

8	WAIT_WRITE_RESP	ro	wo	0x0	Top of Command FIFO - WAIT_WRITE_RESP resetsignal : SW_Reset
7	NATIVE_AXI	ro	wo	0x0	Top of Command FIFO - NATIVE_AXI resetsignal : SW_Reset
6	FW_FLOW_CONTROL	ro	wo	0x0	Top of Command FIFO - FW_FLOW_CONTROL flag resetsignal : SW_Reset
5	CMD_COMPLETE	ro	wo	0x0	Top of Command FIFO - CMD_COMPLETE flag resetsignal : SW_Reset
4:3	SRC_TYPE	ro	wo	0x0	Top of Command FIFO - see CMD FIFO MODE register

enum:SRC_TYPE_e

Name	Value	Description
SRC_MEM	0	Memory - No Flow Control Bus width maximized
SRC_FIFO_8	1	FIFO Byte Mode - Peripheral RX Flow Control.

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>SRC data comes from an 8-bit RX FIFO</td></tr>
<tr><td>SRC_FIFO_32</td><td>2</td><td>AES Output FIFO Flow Control, FIFO Mode transfers - SRC = AES_DATA_OUT</td></tr>
<tr><td>SRC_FILL</td><td>3</td><td>FILL Mode - SRC data comes from DMA_FILL_VALUE register (DMA_FILL_VALUE is not part of DMA CMD)</td></tr>
</table>
encode : SRC_TYPE_e
resetsignal : SW_Reset
</td></tr>
<tr><td>2:0</td><td>DST_TYPE</td><td>ro</td><td>wo</td><td>0x0</td><td>
Top of Command FIFO - see CMD FIFO MODE register

enum:DST_TYPE_e
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>DST_MEM</td><td>0</td><td>Memory - No Flow Control Bus width maximized</td></tr>
<tr><td>DST_FIFO_8</td><td>1</td><td>FIFO Byte Mode - Peripheral TX flow control. DST data goes to an 8-bit TX FIFO</td></tr>
<tr><td>DST_FIFO_32</td><td>2</td><td>Input FIFO Flow Control, FIFO Mode</td></tr>
<tr><td>DST_FIFO_SHA_DIGEST</td><td>3</td><td>SHA Input FIFO Flow Control, FIFO Mode - DST = SHA_MSG/SHA_GEN_DIGEST</td></tr>
<tr><td>DST_FIFO_SHA_CONTEXT</td><td>4</td><td>SHA Input FIFO Flow Control, FIFO Mode - DST = SHA_MSG/SHA_GEN_CONTEXT</td></tr>
<tr><td>DST_FILL_VERIFY</td><td>5</td><td>No Destination. Data are compared with DMA_FILL_VALUE. BYTE LEN must be multiple of 4</td></tr>
</table>
encode : DST_TYPE_e
resetsignal : SW_Reset
</td></tr>
</table>

1.16.1.24 DEBUG

0x000E005C

Debug signals - provided for debug only. Useful if the DMA status is stuck
rtl.reg_enb : false

bits	name	s/w	h/w	default	description
23:16	pop_avail	ro	wo	0x0	resetsignal : SW_Reset
15:8	push_avail	ro	wo	0x0	resetsignal : SW_Reset
7	dst_req	ro	wo	0x0	resetsignal : SW_Reset
6:4	dst_state	ro	wo	0x0	resetsignal : SW_Reset
3	src_req	ro	wo	0x0	resetsignal : SW_Reset
2:0	src_state	ro	wo	0x0	resetsignal : SW_Reset

1.16.1.25 SCRATCH

0x000E0060

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
31:0	DATA	rw	na	0x0	resetsignal : SW_Reset

1.16.1.26 BUS_CTRL

0x000E0064

rtl.reg_enb : false

bits	name	s/w	h/w	default	description
23:20	HPROT_SRC	rw	ro	0x7	Source HPROT for AHB DMA Masters Only set when DMA is IDLE and CMD FIFO is empty. Default is non-cacheable bufferable privileged data. The field is ignored for AXI DMA Masters resetsignal : SW_Reset
19:16	HPROT_DST	rw	ro	0x7	Destination HPROT for AHB DMA Master Only set when DMA is IDLE and CMD FIFO is empty. Default is non-cacheable bufferable privileged data The field is ignored for AXI DMA Masters resetsignal : SW_Reset
15:12	AWCACHE	rw	ro	0x3	Destination AWCACHE for AXI DMA Master Only set when DMA is IDLE and CMD FIFO is empty. Default is normal non-cacheable bufferable. The field is ignored for AHB DMA Masters resetsignal : SW_Reset
11:8	ARCACHE	rw	ro	0x3	Source ARCACHE for AXI DMA Master Only set when DMA is IDLE and CMD FIFO is empty. Default is normal non-cacheable bufferable. The field is ignored for AHB DMA Masters resetsignal : SW_Reset
6:4	AWPROT	rw	ro	0x1	Destination AWPROT for AXI DMA Master Only set when DMA is IDLE and CMD FIFO is empty. Default indicates privileged-secure-data. The field is ignored for AHB DMA Masters resetsignal : SW_Reset
2:0	ARPROT	rw	ro	0x1	Source ARPROT for AXI DMA Master Only set when DMA is IDLE and CMD FIFO is empty. Default indicates privileged-secure-data. The field is ignored for AHB DMA Masters resetsignal : SW_Reset

End RegGroup

End RegGroup

1.17 DW_APB_I2C

0x000F0000 -
0x000F00FF

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000400
chapter : 1.32, Security Subsystem, CS DW I2C
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.17.1 DW_apb_i2c_mem_map_DW_apb_i2c_addr_block1

RegGrp 0x000F0000 - 0x000F00FF

1.17.1.1 IC_CON

Reg. 0x000F0000

bits	name	s/w	h/w	default	description
0	MASTER_MODE	rw	rw	0x1	
2:1	SPEED	rw	rw	0x3	
3	IC_10BITADDR_SLAVE	rw	rw	0x1	
4	IC_10BITADDR_MASTER_rd_only	ro	rw	0x1	
5	IC_RESTART_EN	rw	rw	0x1	
6	IC_SLAVE_DISABLE	rw	rw	0x1	
7	STOP_DET_IFADDRESSED	rw	rw	0x0	
8	TX_EMPTY_CTRL	rw	rw	0x0	
9	RX_FIFO_FULL_HLD_CTRL	rw	rw	0x0	
10	STOP_DET_IF_MASTER_ACTIVE	rw	rw	0x0	
11	BUS_CLEAR_FEATURE_CTRL	rw	rw	0x0	
15:12	RSVD_IC_CON_1	ro	rw	0x0	
16	RSVD_OPTIONAL_SAR_CTRL	ro	rw	0x0	
17	RSVD_SMBUS_SLAVE_QUICK_EN	ro	rw	0x0	
18	RSVD_SMBUS_ARP_EN	ro	rw	0x0	
19	RSVD_SMBUS_PERSISTENT_SLV_ADDR_EN	ro	rw	0x0	
20	RSVD_SMBUS_PERSISTENT_SLV_ADDR2_EN	ro	rw	0x0	
21	RSVD_SMBUS_PERSISTENT_SLV_ADDR3_EN	ro	rw	0x0	
22	RSVD_SMBUS_PERSISTENT_SLV_ADDR4_EN	ro	rw	0x0	
23	RSVD_IC_SAR2_SMBUS_ARP_EN	ro	rw	0x0	
24	RSVD_IC_SAR3_SMBUS_ARP_EN	ro	rw	0x0	
25	RSVD_IC_SAR4_SMBUS_ARP_EN	ro	rw	0x0	
31:26	RSVD_IC_CON_2	ro	rw	0x0	

1.17.1.2 IC_TAR

Reg. 0x000F0004

bits	name	s/w	h/w	default	description
9:0	IC_TAR	rw	rw	0x55	

10	GC_OR_START	rw	rw	0x0	
11	SPECIAL	rw	rw	0x0	
12	IC_10BITADDR_MASTER	rw	rw	0x1	
13	DEVICE_ID	rw	rw	0x0	
15:14	RSVD_IC_TAR_1	ro	rw	0x0	
16	RSVD_SMBUS_QUICK_CMD	ro	rw	0x0	
31:17	RSVD_IC_TAR_2	ro	rw	0x0	

1.17.1.3 IC_SAR

Reg. 0x000F0008

bits	name	s/w	h/w	default	description
9:0	IC_SAR	rw	rw	0x55	volatile : 1
31:10	RSVD_IC_SAR	ro	rw	0x0	volatile : 1

1.17.1.4 IC_HS_MADDR

Reg. 0x000F000C

bits	name	s/w	h/w	default	description
2:0	IC_HS_MAR	rw	rw	0x1	
31:3	RSVD_IC_HS_MAR	ro	rw	0x0	

1.17.1.5 IC_DATA_CMD

Reg. 0x000F0010

bits	name	s/w	h/w	default	description
7:0	DAT	rw	rw	0x0	volatile : 1
8	CMD	wo	rw	0x0	volatile : 1
9	STOP	wo	rw	0x0	volatile : 1
10	RESTART	wo	rw	0x0	volatile : 1
11	FIRST_DATA_BYTE	ro	rw	0x0	volatile : 1
31:12	RSVD_IC_DATA_CMD	ro	rw	0x0	volatile : 1

1.17.1.6 IC_SS_SCL_HCNT

Reg. 0x000F0014

bits	name	s/w	h/w	default	description
15:0	IC_SS_SCL_HCNT	rw	rw	0x190	
31:16	RSVD_IC_SS_SCL_HIGH_COUNT	ro	rw	0x0	

1.17.1.7 IC_SS_SCL_LCNT

Reg. 0x000F0018

bits	name	s/w	h/w	default	description
15:0	IC_SS_SCL_LCNT	rw	rw	0x1D6	
31:16	RSVD_IC_SS_SCL_LOW_COUNT	ro	rw	0x0	

1.17.1.8 IC_FS_SCL_HCNT

Reg. 0x000F001C

bits	name	s/w	h/w	default	description
15:0	IC_FS_SCL_HCNT	rw	rw	0x3C	
31:16	RSVD_IC_FS_SCL_HCNT	ro	rw	0x0	

1.17.1.9 IC_FS_SCL_LCNT

Reg. 0x000F0020

bits	name	s/w	h/w	default	description
15:0	IC_FS_SCL_LCNT	rw	rw	0x82	
31:16	RSVD_IC_FS_SCL_LCNT	ro	rw	0x0	

1.17.1.10 IC_HS_SCL_HCNT

Reg. 0x000F0024

bits	name	s/w	h/w	default	description
15:0	IC_HS_SCL_HCNT	rw	rw	0x6	
31:16	RSVD_IC_HS_SCL_HCNT	ro	rw	0x0	

1.17.1.11 IC_HS_SCL_LCNT

Reg. 0x000F0028

bits	name	s/w	h/w	default	description
15:0	IC_HS_SCL_LCNT	rw	rw	0x10	
31:16	RSVD_IC_HS_SCL_LOW_CNT	ro	rw	0x0	

1.17.1.12 IC_INTR_STAT

Reg. 0x000F002C

bits	name	s/w	h/w	default	description
0	R_RX_UNDER	ro	rw	0x0	volatile : 1
1	R_RX_OVER	ro	rw	0x0	volatile : 1
2	R_RX_FULL	ro	rw	0x0	volatile : 1
3	R_TX_OVER	ro	rw	0x0	volatile : 1
4	R_TX_EMPTY	ro	rw	0x0	volatile : 1
5	R_RD_REQ	ro	rw	0x0	volatile : 1
6	R_TX_ABRT	ro	rw	0x0	volatile : 1
7	R_RX_DONE	ro	rw	0x0	volatile : 1
8	R_ACTIVITY	ro	rw	0x0	volatile : 1
9	R_STOP_DET	ro	rw	0x0	volatile : 1
10	R_START_DET	ro	rw	0x0	volatile : 1
11	R_GEN_CALL	ro	rw	0x0	volatile : 1
12	R_RESTART_DET	ro	rw	0x0	volatile : 1
13	R_MASTER_ON_HOLD	ro	rw	0x0	volatile : 1
14	R_SCL_STUCK_AT_LOW	ro	rw	0x0	volatile : 1
15	RSVD_R_WR_REQ	ro	rw	0x0	volatile : 1
16	RSVD_R_SLV_ADDR1_TAG	ro	rw	0x0	volatile : 1
17	RSVD_R_SLV_ADDR2_TAG	ro	rw	0x0	volatile : 1
18	RSVD_R_SLV_ADDR3_TAG	ro	rw	0x0	volatile : 1
19	RSVD_R_SLV_ADDR4_TAG	ro	rw	0x0	volatile : 1
31:20	RSVD_IC_INTR_STAT	ro	rw	0x0	volatile : 1

1.17.1.13 IC_INTR_MASK

Reg. 0x000F0030

bits	name	s/w	h/w	default	description
0	M_RX_UNDER	rw	rw	0x1	
1	M_RX_OVER	rw	rw	0x1	
2	M_RX_FULL	rw	rw	0x1	
3	M_TX_OVER	rw	rw	0x1	
4	M_TX_EMPTY	rw	rw	0x1	

5	M_RD_REQ	rw	rw	0x1	
6	M_TX_ABRT	rw	rw	0x1	
7	M_RX_DONE	rw	rw	0x1	
8	M_ACTIVITY	rw	rw	0x0	
9	M_STOP_DET	rw	rw	0x0	
10	M_START_DET	rw	rw	0x0	
11	M_GEN_CALL	rw	rw	0x1	
12	M_RESTART_DET	rw	rw	0x0	
13	M_MASTER_ON_HOLD	rw	rw	0x0	
14	M_SCL_STUCK_AT_LOW	rw	rw	0x1	
15	RSVD_M_WR_REQ	ro	rw	0x0	
16	RSVD_M_SLV_ADDR1_TAG	ro	rw	0x0	
17	RSVD_M_SLV_ADDR2_TAG	ro	rw	0x0	
18	RSVD_M_SLV_ADDR3_TAG	ro	rw	0x0	
19	RSVD_M_SLV_ADDR4_TAG	ro	rw	0x0	
31:20	RSVD_IC_INTR_STAT	ro	rw	0x0	

1.17.1.14 IC_RAW_INTR_STAT

0x000F0034

bits	name	s/w	h/w	default	description
0	RX_UNDER	ro	rw	0x0	volatile : 1
1	RX_OVER	ro	rw	0x0	volatile : 1
2	RX_FULL	ro	rw	0x0	volatile : 1
3	TX_OVER	ro	rw	0x0	volatile : 1
4	TX_EMPTY	ro	rw	0x0	volatile : 1
5	RD_REQ	ro	rw	0x0	volatile : 1
6	TX_ABRT	ro	rw	0x0	volatile : 1
7	RX_DONE	ro	rw	0x0	volatile : 1
8	ACTIVITY	ro	rw	0x0	volatile : 1
9	STOP_DET	ro	rw	0x0	volatile : 1
10	START_DET	ro	rw	0x0	volatile : 1
11	GEN_CALL	ro	rw	0x0	volatile : 1
12	RESTART_DET	ro	rw	0x0	volatile : 1
13	MASTER_ON_HOLD	ro	rw	0x0	volatile : 1
14	SCL_STUCK_AT_LOW	ro	rw	0x0	volatile : 1
15	RSVD_WR_REQ	ro	rw	0x0	volatile : 1
16	RSVD_SLV_ADDR1_TAG	ro	rw	0x0	volatile : 1
17	RSVD_SLV_ADDR2_TAG	ro	rw	0x0	volatile : 1
18	RSVD_SLV_ADDR3_TAG	ro	rw	0x0	volatile : 1
19	RSVD_SLV_ADDR4_TAG	ro	rw	0x0	volatile : 1
31:20	RSVD_IC_RAW_INTR_STAT	ro	rw	0x0	volatile : 1

1.17.1.15 IC_RX_TL

0x000F0038

bits	name	s/w	h/w	default	description
7:0	RX_TL	rw	rw	0x0	
31:8	RSVD_IC_RX_TL	ro	rw	0x0	

1.17.1.16 IC_TX_TL

0x000F003C

bits	name	s/w	h/w	default	description
7:0	TX_TL	rw	rw	0x0	
31:8	RSVD_IC_TX_TL	ro	rw	0x0	

1.17.1.17 IC_CLR_INTR

Reg. 0x000F0040

bits	name	s/w	h/w	default	description
0	CLR_INTR	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_INTR	ro	rw	0x0	volatile : 1

1.17.1.18 IC_CLR_RX_UNDER

Reg. 0x000F0044

bits	name	s/w	h/w	default	description
0	CLR_RX_UNDER	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_RX_UNDER	ro	rw	0x0	volatile : 1

1.17.1.19 IC_CLR_RX_OVER

Reg. 0x000F0048

bits	name	s/w	h/w	default	description
0	CLR_RX_OVER	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_RX_OVER	ro	rw	0x0	volatile : 1

1.17.1.20 IC_CLR_TX_OVER

Reg. 0x000F004C

bits	name	s/w	h/w	default	description
0	CLR_TX_OVER	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_TX_OVER	ro	rw	0x0	volatile : 1

1.17.1.21 IC_CLR_RD_REQ

Reg. 0x000F0050

bits	name	s/w	h/w	default	description
0	CLR_RD_REQ	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_RD_REQ	ro	rw	0x0	volatile : 1

1.17.1.22 IC_CLR_TX_ABRT

Reg. 0x000F0054

bits	name	s/w	h/w	default	description
0	CLR_TX_ABRT	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_TX_ABRT	ro	rw	0x0	volatile : 1

1.17.1.23 IC_CLR_RX_DONE

Reg. 0x000F0058

bits	name	s/w	h/w	default	description
0	CLR_RX_DONE	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_RX_DONE	ro	rw	0x0	volatile : 1

1.17.1.24 IC_CLR_ACTIVITY

Reg. 0x000F005C

bits	name	s/w	h/w	default	description
0	CLR_ACTIVITY	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_ACTIVITY	ro	rw	0x0	volatile : 1

1.17.1.25 IC_CLR_STOP_DET

Reg. 0x000F0060

bits	name	s/w	h/w	default	description
0	CLR_STOP_DET	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_STOP_DET	ro	rw	0x0	volatile : 1

1.17.1.26 IC_CLR_START_DET

Reg. 0x000F0064

bits	name	s/w	h/w	default	description
0	CLR_START_DET	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_START_DET	ro	rw	0x0	volatile : 1

1.17.1.27 IC_CLR_GEN_CALL

Reg. 0x000F0068

bits	name	s/w	h/w	default	description
0	CLR_GEN_CALL	ro	rw	0x0	volatile : 1
31:1	RSVD_IC_CLR_GEN_CALL	ro	rw	0x0	volatile : 1

1.17.1.28 IC_ENABLE

Reg. 0x000F006C

bits	name	s/w	h/w	default	description
0	ENABLE	rw	rw	0x0	
1	ABORT	rw	rw	0x0	
2	TX_CMD_BLOCK	rw	rw	0x1	
3	SDA_STUCK_RECOVERY_ENABLE	rw	rw	0x0	
15:4	RSVD_IC_ENABLE_1	ro	rw	0x0	
16	RSVD_SMBUS_CLK_RESET	ro	rw	0x0	
17	RSVD_SMBUS_SUSPEND_EN	ro	rw	0x0	
18	RSVD_SMBUS_ALERT_EN	ro	rw	0x0	
19	RSVD_IC_SAR_EN	ro	rw	0x0	
20	RSVD_IC_SAR2_EN	ro	rw	0x0	
21	RSVD_IC_SAR3_EN	ro	rw	0x0	
22	RSVD_IC_SAR4_EN	ro	rw	0x0	
31:23	RSVD_IC_ENABLE_2	ro	rw	0x0	

1.17.1.29 IC_STATUS

Reg. 0x000F0070

bits	name	s/w	h/w	default	description
0	ACTIVITY	ro	rw	0x0	volatile : 1
1	TFNF	ro	rw	0x1	volatile : 1
2	TFE	ro	rw	0x1	volatile : 1
3	RFNE	ro	rw	0x0	volatile : 1

4	RFF	ro	rw	0x0	volatile : 1
5	MST_ACTIVITY	ro	rw	0x0	volatile : 1
6	SLV_ACTIVITY	ro	rw	0x0	volatile : 1
7	MST_HOLD_TX_FIFO_EMPTY	ro	rw	0x0	volatile : 1
8	MST_HOLD_RX_FIFO_FULL	ro	rw	0x0	volatile : 1
9	SLV_HOLD_TX_FIFO_EMPTY	ro	rw	0x0	volatile : 1
10	SLV_HOLD_RX_FIFO_FULL	ro	rw	0x0	volatile : 1
11	SDA_STUCK_NOT_RECOVERED	ro	rw	0x0	volatile : 1
12	RSVD_SLV_ISO_SAR_DATA_CLK_STRETCH	ro	rw	0x0	volatile : 1
15:13	RSVD_IC_STATUS_1	ro	rw	0x0	volatile : 1
16	RSVD_SMBUS_QUICK_CMD_BIT	ro	rw	0x0	volatile : 1
17	RSVD_SMBUS_SLAVE_ADDR_VALID	ro	rw	0x0	volatile : 1
18	RSVD_SMBUS_SLAVE_ADDR_RESOLVED	ro	rw	0x0	volatile : 1
19	RSVD_SMBUS_SUSPEND_STATUS	ro	rw	0x0	volatile : 1
20	RSVD_SMBUS_ALERT_STATUS	ro	rw	0x0	volatile : 1
21	RSVD_SMBUS_SLAVE_ADDR2_VALID	ro	rw	0x0	volatile : 1
22	RSVD_SMBUS_SLAVE_ADDR2_RESOLVED	ro	rw	0x0	volatile : 1
23	RSVD_SMBUS_SLAVE_ADDR3_VALID	ro	rw	0x0	volatile : 1
24	RSVD_SMBUS_SLAVE_ADDR3_RESOLVED	ro	rw	0x0	volatile : 1
25	RSVD_SMBUS_SLAVE_ADDR4_VALID	ro	rw	0x0	volatile : 1
26	RSVD_SMBUS_SLAVE_ADDR4_RESOLVED	ro	rw	0x0	volatile : 1
31:27	RSVD_IC_STATUS_2	ro	rw	0x0	volatile : 1

1.17.1.30 IC_TXFLR

Reg. 0x000F0074

bits	name	s/w	h/w	default	description
4:0	TXFLR	ro	rw	0x0	volatile : 1
31:5	RSVD_TXFLR	ro	rw	0x0	volatile : 1

1.17.1.31 IC_RXFLR

Reg. 0x000F0078

bits	name	s/w	h/w	default	description
4:0	RXFLR	ro	rw	0x0	volatile : 1
31:5	RSVD_RXFLR	ro	rw	0x0	volatile : 1

1.17.1.32 IC_SDA_HOLD

Reg. 0x000F007C

bits	name	s/w	h/w	default	description
15:0	IC_SDA_TX_HOLD	rw	rw	0x1	

23:16	IC_SDA_RX_HOLD	rw	rw	0x0	
31:24	RSVD_IC_SDA_HOLD	ro	rw	0x0	

1.17.1.33 IC_TX_ABRT_SOURCE

Reg. 0x000F0080

bits	name	s/w	h/w	default	description
0	ABRT_7B_ADDR_NOACK	ro	rw	0x0	volatile : 1
1	ABRT_10ADDR1_NOACK	ro	rw	0x0	volatile : 1
2	ABRT_10ADDR2_NOACK	ro	rw	0x0	volatile : 1
3	ABRT_TXDATA_NOACK	ro	rw	0x0	volatile : 1
4	ABRT_GCALL_NOACK	ro	rw	0x0	volatile : 1
5	ABRT_GCALL_READ	ro	rw	0x0	volatile : 1
6	ABRT_HS_ACKDET	ro	rw	0x0	volatile : 1
7	ABRT_SBYTE_ACKDET	ro	rw	0x0	volatile : 1
8	ABRT_HS_NORSTRT	ro	rw	0x0	volatile : 1
9	ABRT_SBYTE_NORSTRT	ro	rw	0x0	volatile : 1
10	ABRT_10B_RD_NORSTRT	ro	rw	0x0	volatile : 1
11	ABRT_MASTER_DIS	ro	rw	0x0	volatile : 1
12	ARB_LOST	ro	rw	0x0	volatile : 1
13	ABRT_SLVFLUSH_TXFIFO	ro	rw	0x0	volatile : 1
14	ABRT_SLV_ARBLOST	ro	rw	0x0	volatile : 1
15	ABRT_SLVRD_INTX	ro	rw	0x0	volatile : 1
16	ABRT_USER_ABRT	ro	rw	0x0	volatile : 1
17	ABRT_SDA_STUCK_AT_LOW	ro	rw	0x0	volatile : 1
18	ABRT_DEVICE_NOACK	ro	rw	0x0	volatile : 1
19	ABRT_DEVICE_SLVADDR_NOACK	ro	rw	0x0	volatile : 1
20	ABRT_DEVICE_WRITE	ro	rw	0x0	volatile : 1
22:21	RSVD_IC_TX_ABRT_SOURCE	ro	rw	0x0	volatile : 1
31:23	TX_FLUSH_CNT	ro	rw	0x0	volatile : 1

1.17.1.34 IC_SLV_DATA_NACK_ONLY

Reg. 0x000F0084

bits	name	s/w	h/w	default	description
0	NACK	rw	rw	0x0	
31:1	RSVD_IC_SLV_DATA_NACK_ONLY	ro	rw	0x0	

1.17.1.35 IC_DMA_CR

Reg. 0x000F0088

bits	name	s/w	h/w	default	description
0	RDMAE	rw	rw	0x0	
1	TDMAE	rw	rw	0x0	
31:2	RSVD_IC_DMA_CR_2_31	ro	rw	0x0	

1.17.1.36 IC_DMA_TDLR

0x000F008C

bits	name	s/w	h/w	default	description
3:0	DMATDL	rw	rw	0x0	
31:4	RSVD_DMA_TDLR	ro	rw	0x0	

1.17.1.37 IC_DMA_RDLR

0x000F0090

bits	name	s/w	h/w	default	description
3:0	DMARDL	rw	rw	0x0	
31:4	RSVD_DMA_RDLR	ro	rw	0x0	

1.17.1.38 IC_SDA_SETUP

0x000F0094

bits	name	s/w	h/w	default	description
7:0	SDA_SETUP	rw	rw	0x64	
31:8	RSVD_IC_SDA_SETUP	ro	rw	0x0	

1.17.1.39 IC_ACK_GENERAL_CALL

0x000F0098

bits	name	s/w	h/w	default	description
0	ACK_GEN_CALL	rw	rw	0x1	
31:1	RSVD_IC_ACK_GEN_1_31	ro	rw	0x0	

1.17.1.40 IC_ENABLE_STATUS

0x000F009C

bits	name	s/w	h/w	default	description
0	IC_EN	ro	rw	0x0	volatile : 1
1	SLV_DISABLED_WHILE_BUSY	ro	rw	0x0	volatile : 1
2	SLV_RX_DATA_LOST	ro	rw	0x0	volatile : 1
31:3	RSVD_IC_ENABLE_STATUS	ro	rw	0x0	volatile : 1

1.17.1.41 IC_FS_SPKLEN

0x000F00A0

bits	name	s/w	h/w	default	description
7:0	IC_FS_SPKLEN	rw	rw	0x5	
31:8	RSVD_IC_FS_SPKLEN	ro	rw	0x0	

1.17.1.42 IC_HS_SPKLEN

0x000F00A4

bits	name	s/w	h/w	default	description
7:0	IC_HS_SPKLEN	rw	rw	0x1	
31:8	RSVD_IC_HS_SPKLEN	ro	rw	0x0	

1.17.1.43 IC_CLR_RESTART_DET

0x000F00A8

bits	name	s/w	h/w	default	description
0	CLR_RESTART_DET	ro	rw	0x0	volatile : 1

31:1	RSVD_IC_CLR_RESTART_DET	ro	rw	0x0	volatile : 1

1.17.1.44 IC_SCL_STUCK_AT_LOW_TIMEOUT

0x000F00AC

bits	name	s/w	h/w	default	description
31:0	IC_SCL_STUCK_LOW_TIMEOUT	rw	rw	0xFFFFFFFF	

1.17.1.45 IC_SDA_STUCK_AT_LOW_TIMEOUT

0x000F00B0

bits	name	s/w	h/w	default	description
31:0	IC_SDA_STUCK_LOW_TIMEOUT	rw	rw	0xFFFFFFFF	

1.17.1.46 IC_CLR_SCL_STUCK_DET

0x000F00B4

bits	name	s/w	h/w	default	description
0	CLR_SCL_STUCK_DET	ro	rw	0x0	volatile : 1
31:1	RSVD_CLR_SCL_STUCK_DET	ro	rw	0x0	volatile : 1

1.17.1.47 IC_DEVICE_ID

0x000F00B8

bits	name	s/w	h/w	default	description
23:0	DEVICE_ID	ro	rw	0x0	
31:24	RSVD_IC_DEVICE_ID	ro	rw	0x0	

1.17.1.48 REG_TIMEOUT_RST

0x000F00F0

bits	name	s/w	h/w	default	description
3:0	REG_TIMEOUT_RST_rw	rw	rw	0x8	volatile : 1
31:4	RSVD_REG_TIMEOUT_RST	ro	rw	0x0	volatile : 1

1.17.1.49 IC_COMP_PARAM_1

0x000F00F4

bits	name	s/w	h/w	default	description
1:0	APB_DATA_WIDTH	ro	rw	0x2	
3:2	MAX_SPEED_MODE	ro	rw	0x3	
4	HC_COUNT_VALUES	ro	rw	0x0	
5	INTR_IO	ro	rw	0x1	
6	HAS_DMA	ro	rw	0x1	
7	ADD_ENCODED_PARAMS	ro	rw	0x1	
15:8	RX_BUFFER_DEPTH	ro	rw	0xF	
23:16	TX_BUFFER_DEPTH	ro	rw	0xF	
31:24	RSVD_IC_COMP_PARAM_1	ro	rw	0x0	

1.17.1.50 IC_COMP_VERSION

0x000F00F8

bits	name	s/w	h/w	default	description
31:0	IC_COMP_VERSION	ro	rw	0x3230332A	

1.17.1.51 IC_COMP_TYPE

Reg. 0x000F00FC

bits	name	s/w	h/w	default	description
31:0	IC_COMP_TYPE	ro	rw	0x44570140	

End RegGroup

End RegGroup

1.18 DW_APB_UART

RegGrp 0x000F1000 - 0x000F10FF

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000400
chapter : 1.33, Security Subsystem, CS DW UART
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.18.1 uart_memory_map_uart_address_block

RegGrp 0x000F1000 - 0x000F10FF

1.18.1.1 RBR

Reg. 0x000F1000

bits	name	s/w	h/w	default	description
8:0	RBR	ro	rw	0x0	
31:9	RSVD_RBR	ro	rw	0x0	

1.18.1.2 IER

Reg. 0x000F1004

bits	name	s/w	h/w	default	description
0	ERBFI	rw	rw	0x0	
1	ETBEI	rw	rw	0x0	
2	ELSI	rw	rw	0x0	
3	EDSSI	rw	rw	0x0	
4	ELCOLR	ro	rw	0x0	
6:5	RSVD_IER_6to5	ro	rw	0x0	
7	PTIME	rw	rw	0x0	
31:8	RSVD_IER_31to8	ro	rw	0x0	

1.18.1.3 IIR

Reg. 0x000F1008

bits	name	s/w	h/w	default	description
3:0	IID	ro	rw	0x1	
5:4	RSVD_IIR_5to4	ro	rw	0x0	
7:6	FIFOSE	ro	rw	0x0	
31:8	RSVD_IIR_31to8	ro	rw	0x0	

1.18.1.4 LCR

Reg. 0x000F100C

bits	name	s/w	h/w	default	description
1:0	DLS	rw	rw	0x0	
2	STOP	rw	rw	0x0	
3	PEN	rw	rw	0x0	
4	EPS	rw	rw	0x0	
5	SP	rw	rw	0x0	
6	BC	rw	rw	0x0	
7	DLAB	rw	rw	0x0	
31:8	RSVD_LCR_31to8	ro	rw	0x0	

1.18.1.5 MCR

Reg. 0x000F1010

bits	name	s/w	h/w	default	description
0	DTR	rw	rw	0x0	
1	RTS	rw	rw	0x0	
2	OUT1	rw	rw	0x0	
3	OUT2	rw	rw	0x0	
4	LoopBack	rw	rw	0x0	
5	AFCE	rw	rw	0x0	
6	SIRE	ro	rw	0x0	
31:7	RSVD_MCR_31to7	ro	rw	0x0	

1.18.1.6 LSR

Reg. 0x000F1014

bits	name	s/w	h/w	default	description
0	DR	ro	rw	0x0	
1	OE	ro	rw	0x0	
2	PE	ro	rw	0x0	
3	FE	ro	rw	0x0	
4	BI	ro	rw	0x0	
5	THRE	ro	rw	0x1	
6	TEMT	ro	rw	0x1	
7	RFE	ro	rw	0x0	
8	ADDR_RCVD	ro	rw	0x0	
31:9	RSVD_LSR_31to9	ro	rw	0x0	

1.18.1.7 MSR

Reg. 0x000F1018

bits	name	s/w	h/w	default	description
0	DCTS	ro	rw	0x0	
1	DDSR	ro	rw	0x0	
2	TERI	ro	rw	0x0	
3	DDCD	ro	rw	0x0	
4	CTS	ro	rw	0x0	
5	DSR	ro	rw	0x0	
6	RI	ro	rw	0x0	
7	DCD	ro	rw	0x0	
31:8	RSVD_MSR_31to8	ro	rw	0x0	

1.18.1.8 SCR

Reg. 0x000F101C

bits	name	s/w	h/w	default	description
7:0	SCR	rw	rw	0x0	
31:8	RSVD_SCR_31to8	ro	rw	0x0	

1.18.1.9 FAR

Reg. 0x000F1070

bits	name	s/w	h/w	default	description

0	FAR	rw	rw	0x0	
31:1	RSVD_FAR_31to1	ro	rw	0x0	

1.18.1.10 TFR

Reg. 0x000F1074

bits	name	s/w	h/w	default	description
7:0	TFR	ro	rw	0x0	
31:8	RSVD_TFR_31to8	ro	rw	0x0	

1.18.1.11 RFW

Reg. 0x000F1078

bits	name	s/w	h/w	default	description
7:0	RFWD	wo	rw	0x0	
8	RFPE	wo	rw	0x0	
9	RFFE	wo	rw	0x0	
31:10	RSVD_RFW_31to10	ro	rw	0x0	

1.18.1.12 USR

Reg. 0x000F107C

bits	name	s/w	h/w	default	description
0	RSVD_BUSY	ro	rw	0x0	volatile : 1
1	RSVD_TFNF	ro	rw	0x0	volatile : 1
2	RSVD_TFE	ro	rw	0x0	volatile : 1
3	RSVD_RFNE	ro	rw	0x0	volatile : 1
4	RSVD_RFF	ro	rw	0x0	volatile : 1
31:5	RSVD_USR_31to5	ro	rw	0x0	volatile : 1

1.18.1.13 HTX

Reg. 0x000F10A4

bits	name	s/w	h/w	default	description
0	HTX	rw	rw	0x0	
31:1	RSVD_HTX_31to1	ro	rw	0x0	

1.18.1.14 DMASA

Reg. 0x000F10A8

bits	name	s/w	h/w	default	description
0	DMASA	wo	rw	0x0	
31:1	RSVD_DMASA_31to1	ro	rw	0x0	

1.18.1.15 RAR

Reg. 0x000F10C4

bits	name	s/w	h/w	default	description
7:0	RAR	rw	rw	0x0	
31:8	RSVD_RAR_31to8	ro	rw	0x0	

1.18.1.16 TAR

Reg. 0x000F10C8

bits	name	s/w	h/w	default	description
7:0	TAR	rw	rw	0x0	
31:8	RSVD_TAR_31to8	ro	rw	0x0	

1.18.1.17 LCR_EXT

Reg. 0x000F10CC

bits	name	s/w	h/w	default	description
0	DLS_E	rw	rw	0x0	volatile : 1
1	ADDR_MATCH	rw	rw	0x0	volatile : 1
2	SEND_ADDR	rw	rw	0x0	volatile : 1
3	TRANSMIT_MODE	rw	rw	0x0	volatile : 1
31:4	RSVD_LCR_EXT	ro	rw	0x0	volatile : 1

1.18.1.18 UART_PROT_LEVEL

Reg. 0x000F10D0

bits	name	s/w	h/w	default	description
2:0	UART_PROT_LEVEL	rw	rw	0x2	
31:3	RSVD_UART_PROT_LEVEL	ro	rw	0x0	

1.18.1.19 REG_TIMEOUT_RST

Reg. 0x000F10D4

bits	name	s/w	h/w	default	description
3:0	REG_TIMEOUT_RST	rw	rw	0x8	volatile : 1
31:4	RSVD_REG_TIMEOUT_RST	ro	rw	0x0	volatile : 1

1.18.1.20 CPR

Reg. 0x000F10F4

bits	name	s/w	h/w	default	description
1:0	APB_DATA_WIDTH	ro	rw	0x2	
3:2	RSVD_CPR_3to2	ro	rw	0x0	
4	AFCE_MODE	ro	rw	0x1	
5	THRE_MODE	ro	rw	0x1	
6	SIR_MODE	ro	rw	0x0	
7	SIR_LP_MODE	ro	rw	0x0	
8	ADDITIONAL_FEAT	ro	rw	0x1	
9	FIFO_ACCESS	ro	rw	0x1	
10	FIFO_STAT	ro	rw	0x0	
11	SHADOW	ro	rw	0x0	
12	UART_ADD_ENCODED_PARAMS	ro	rw	0x1	
13	DMA_EXTRA	ro	rw	0x1	
15:14	RSVD_CPR_15to14	ro	rw	0x0	
23:16	FIFO_MODE	ro	rw	0x1	
31:24	RSVD_CPR_31to24	ro	rw	0x0	

1.18.1.21 UCV

Reg. 0x000F10F8

bits	name	s/w	h/w	default	description
31:0	UART_Component_Version	ro	rw	0x3430332A	

1.18.1.22 CTR

Reg. 0x000F10FC

bits	name	s/w	h/w	default	description
31:0	Peripheral_ID	ro	rw	0x44570110	

End RegGroup

End RegGroup

1.19 SPI_REGS_SYS

RegGrp

0x000F2000 - 0x000F20C7

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000400
chapter : 1.35, Security Subsystem, CS SPI
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.19.1 spi_regs

RegGrp

0x000F2000 - 0x000F20C7

1.19.1.1 SPI_CONTROL_REG

Reg.

0x000F2080

bits	name	s/w	h/w	default	description
2:0	A_BSY	rw	rw	0x7	
3	DMA_EN	rw	rw	0x0	
4	SS	ro	rw	0x0	
6	OUT_DIS	rw	rw	0x0	
7	SNF	rw	rw	0x0	
8	CPHA	rw	rw	0x0	
9	CPOL	rw	rw	0x0	
10	DAT_TX	rw	rw	0x0	
11	DAT_RES	wo	rw	0x0	
12	DEB_RES	wo	rw	0x0	
13	SPI_RES	rw	rw	0x1	
14	DEV_PAD	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.2 SPI_MASTER_CONTROL_REG

Reg.

0x000F2084

bits	name	s/w	h/w	default	description
0	MASTER	rw	rw	0x0	
2:1	SS_OC	rw	rw	0x0	
3	SLOW_SS	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.3 SPI_COMMAND_REG

Reg.

0x000F2088

bits	name	s/w	h/w	default	description
7:0	COMMAND	rw	rw	0x0	
15:8	REGISTERS	rw	rw	0x0	
16	RUN	rw	rw	0x0	
17	POLL	rw	rw	0x0	
19:18	RC_TX	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.4 SPI_INTERRUPT_STATUS_REG

Reg.

0x000F208C

bits	name	s/w	h/w	default	description
2	FIFO_E	rw	rw	0x0	
5	TRAN_D	rw	rw	0x0	
11	DA_AEF	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.5 SPI_INTERRUPT_MASK_REG

Reg. 0x000F2090

bits	name	s/w	h/w	default	description
2	FIFO_E	rw	rw	0x0	
5	TRAN_D	rw	rw	0x0	
11	DA_AEF	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.6 SPI_BYTE_COUNT_REG

Reg. 0x000F2094

bits	name	s/w	h/w	default	description
31:0	RESERVED	ro	rw	0x0	

1.19.1.7 SPI_DRQ_COUNT_REG

Reg. 0x000F2098

bits	name	s/w	h/w	default	description
15:0	COUNT	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.8 SPI_DEBUG_REG

Reg. 0x000F209C

bits	name	s/w	h/w	default	description
7:0	RSVD_BYTE0	ro	rw	0x0	
15:8	RSVD_BYTE1	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.9 DATA_PORT_BASE_REG

Reg. 0x000F20A0

bits	name	s/w	h/w	default	description
17:2	BASE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.10 DEBUG_PORT_BASE_REG

Reg. 0x000F20A4

bits	name	s/w	h/w	default	description
17:2	RSVD_BASE	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.11 DEBUG_PORT_WR_PTR_REG

Reg. 0x000F20A8

bits	name	s/w	h/w	default	description
10:0	RSVD_WR_PTR	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.12 DEBUG_PORT_RD_PTR_REG

Reg. 0x000F20AC

bits	name	s/w	h/w	default	description
10:0	RSVD_RD_PTR	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.13 DATA_PORT_DATA_REG

Reg. 0x000F20B0

bits	name	s/w	h/w	default	description
31:0	DATA	rw	rw	0x0	

1.19.1.14 DEBUG_PORT_DATA_REG

Reg. 0x000F20B4

bits	name	s/w	h/w	default	description
31:0	RSVD_DATA	ro	rw	0x0	

1.19.1.15 IRQ_DMARQ_DATA_THRESHOLD_REG

Reg. 0x000F20B8

bits	name	s/w	h/w	default	description
9:0	THRESHOLD	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.16 DEBUG_PORT_SIZE_REG

Reg. 0x000F20BC

bits	name	s/w	h/w	default	description
2:0	RSVD_SIZE	ro	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.17 TX_THRESHOLD_REG

Reg. 0x000F20C0

bits	name	s/w	h/w	default	description
9:0	THRESHOLD	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.19.1.18 DMARQ_DEBUG_THRESHOLD_REG

Reg. 0x000F20C4

bits	name	s/w	h/w	default	description
9:0	THRESHOLD	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

End RegGroup

End RegGroup

1.20 GPIO_SYS

RegGrp 0x000F3000 -
0x000F30AB

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000400
chapter : 1.34, Security Subsystem, CS GPIO
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.20.1 gpio

RegGrp 0x000F3000 -
0x000F30AB

1.20.1.1 GPIO_PER

Reg. 0x000F3000

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x3C3FF	
31	reserved_31	ro	rw	0x0	

1.20.1.2 GPIO_PECR

Reg. 0x000F3004

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.3 GPIO_PESR

Reg. 0x000F3008

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.4 GPIO_OER

Reg. 0x000F3010

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.5 GPIO_OECR

Reg. 0x000F3014

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.6 GPIO_OESR

Reg. 0x000F3018

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.7 GPIO_ODR

Reg. 0x000F3020

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.8 GPIO_ODCR

Reg. 0x000F3024

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.9 GPIO_ODSR

Reg. 0x000F3028

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.10 GPIO_PSR

0x000F3030

bits	name	s/w	h/w	default	description
17:0	VALUE	ro	rw	0x3FFFF	
31	reserved_31	ro	rw	0x0	

1.20.1.11 GPIO_SPR

0x000F3040

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.12 GPIO_SPCR

0x000F3044

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.13 GPIO_SPSR

0x000F3048

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.14 GPIO_IMR

0x000F3050

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.15 GPIO_IMCR

0x000F3054

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.16 GPIO_IMSR

0x000F3058

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.17 GPIO_NIR

0x000F3060

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.18 GPIO_NICR

0x000F3064

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.19 GPIO_NISR

Reg. 0x000F3068

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.20 GPIO_PE

Reg. 0x000F3070

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.21 GPIO_PEC

Reg. 0x000F3074

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.22 GPIO_PES

Reg. 0x000F3078

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.23 GPIO_PS

Reg. 0x000F3080

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.24 GPIO_PSC

Reg. 0x000F3084

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.25 GPIO_PSS

Reg. 0x000F3088

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.26 GPIO_STE

Reg. 0x000F3090

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.27 GPIO_STEC

Reg. 0x000F3094

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.28 GPIO_STES

Reg. 0x000F3098

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.29 GPIO_IER

Reg. 0x000F30A0

bits	name	s/w	h/w	default	description
17:0	VALUE	rw	rw	0x3FFFF	
31	reserved_31	ro	rw	0x0	

1.20.1.30 GPIO_IECR

Reg. 0x000F30A4

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.20.1.31 GPIO_IERS

Reg. 0x000F30A8

bits	name	s/w	h/w	default	description
17:0	VALUE	wo	rw	0x0	
31	reserved_31	ro	rw	0x0	

End RegGroup

End RegGroup

1.21 GPIO_SFR_SYS

RegGrp 0x000F4000 - 0x000F418B

Present for CS SoCs only. Access write-ignored, read zeros for CSSD/HDD/ESSD
decode_size : 0x00000400
chapter : 1.37, Security Subsystem, CS GPIO
blockgroup : SECUREPROCESSOR
revision : revision: 56f92f4

1.21.1 gpio_sfr

RegGrp 0x000F4000 - 0x000F418B

1.21.1.1 GPIO_OE_PA

Reg. 0x000F4000

bits	name	s/w	h/w	default	description

17:0	GPIO_OE_PA	rw	rw	0x1C00	
31	reserved_31	ro	rw	0x0	

1.21.1.2 GPIO_OE_PB

Reg. 0x000F4008

bits	name	s/w	h/w	default	description
17:0	GPIO_OE_PB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.3 GPIO_OUT_PA

Reg. 0x000F4010

bits	name	s/w	h/w	default	description
17:0	GPIO_OUT_PA	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.4 GPIO_OUT_PB

Reg. 0x000F4018

bits	name	s/w	h/w	default	description
17:0	GPIO_OUT_PB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.5 GPIO_IN_PA

Reg. 0x000F4020

bits	name	s/w	h/w	default	description
17:0	GPIO_IN_PA	ro	rw	0x3E3C9	
31	reserved_31	ro	rw	0x0	

1.21.1.6 GPIO_IN_PB

Reg. 0x000F4028

bits	name	s/w	h/w	default	description
17:0	GPIO_IN_PB	ro	rw	0x3FFF0	
31	reserved_31	ro	rw	0x0	

1.21.1.7 GPIO_IE_PA

Reg. 0x000F4030

bits	name	s/w	h/w	default	description
17:0	GPIO_IE_FROM_PA	rw	rw	0x11C00	
31	reserved_31	ro	rw	0x0	

1.21.1.8 GPIO_IE_PB

Reg. 0x000F4038

bits	name	s/w	h/w	default	description
17:0	GPIO_IE_FROM_PB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.9 GPIO_PE_PA

Reg. 0x000F4040

bits	name	s/w	h/w	default	description
17:0	GPIO_PE_FROM_PA	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.10 GPIO_PE_PB

0x000F4048

bits	name	s/w	h/w	default	description
17:0	GPIO_PE_FROM_PB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.11 GPIO_PS_PA

0x000F4050

bits	name	s/w	h/w	default	description
17:0	GPIO_PS_FROM_PA	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.12 GPIO_PS_PB

0x000F4058

bits	name	s/w	h/w	default	description
17:0	GPIO_PS_FROM_PB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.13 ROT_TRACE_EN

0x000F4100

bits	name	s/w	h/w	default	description
0	ROT_SBS_TRACE_EN	rw	rw	0x0	
1	ROT_TEST_MUX_TRACE_EN	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.14 HTU_SBS_FRAME_SEL

0x000F4108

bits	name	s/w	h/w	default	description
0	SVCI2AHB	rw	rw	0x0	
1	AXI2SVCI	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.15 SBS_SVCI2AHB_CONFIG

0x000F4110

bits	name	s/w	h/w	default	description
1:0	TYPE	rw	rw	0x0	
2	INC_EXC	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.16 SBS_SVCI2AHB_BASE

0x000F4118

bits	name	s/w	h/w	default	description
31:16	ADDR	rw	rw	0x0	

1.21.1.17 SBS_SVCI2AHB_OFFSET_0

0x000F4120

bits	name	s/w	h/w	default	description
0	SEL_POL	rw	rw	0x0	
15:3	MASK	rw	rw	0x0	
31:19	ADDR	rw	rw	0x0	

1.21.1.18 SBS_SVCI2AHB_OFFSET_1

Reg. 0x000F4128

bits	name	s/w	h/w	default	description
0	SEL_POL	rw	rw	0x0	
15:3	MASK	rw	rw	0x0	
31:19	ADDR	rw	rw	0x0	

1.21.1.19 SBS_SVCI2AHB_DATA_SEL

Reg. 0x000F4130

bits	name	s/w	h/w	default	description
0	SVCI2AHB	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.20 SBS_AXI2SVCI_CONFIG

Reg. 0x000F4138

bits	name	s/w	h/w	default	description
1:0	TYPE	rw	rw	0x0	
2	INC_EXC	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.21 SBS_AXI2SVCI_BASE

Reg. 0x000F4140

bits	name	s/w	h/w	default	description
31:16	ADDR	rw	rw	0x0	

1.21.1.22 SBS_AXI2SVCI_OFFSET_0

Reg. 0x000F4148

bits	name	s/w	h/w	default	description
0	SEL_POL	rw	rw	0x0	
15:3	MASK	rw	rw	0x0	
31:19	ADDR	rw	rw	0x0	

1.21.1.23 SBS_AXI2SVCI_OFFSET_1

Reg. 0x000F4150

bits	name	s/w	h/w	default	description
0	SEL_POL	rw	rw	0x0	
15:3	MASK	rw	rw	0x0	
31:19	ADDR	rw	rw	0x0	

1.21.1.24 SBS_AXI2SVCI_DATA_SEL

Reg. 0x000F4158

bits	name	s/w	h/w	default	description
0	AXI2SVCI	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.25 HTU_TEST_MUX_MASK_31_0

Reg. 0x000F4160

bits	name	s/w	h/w	default	description
31:0	MASK	rw	rw	0x0	

1.21.1.26 HTU_TEST_MUX_MASK_47_32

Reg. 0x000F4168

bits	name	s/w	h/w	default	description

15:0	MASK	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.27 HTU_SBS_SVCI2AHB_COUNT

Reg. 0x000F4170

bits	name	s/w	h/w	default	description
15:0	COUNT	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.28 HTU_SBS_AXI2SVCI_COUNT

Reg. 0x000F4178

bits	name	s/w	h/w	default	description
15:0	COUNT	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.29 HTU_TEST_MUX_COUNT

Reg. 0x000F4180

bits	name	s/w	h/w	default	description
15:0	COUNT	rw	rw	0x0	
31	reserved_31	ro	rw	0x0	

1.21.1.30 SPI_DMA_REQ

Reg. 0x000F4188

bits	name	s/w	h/w	default	description
4:0	REQUEST	rw	rw	0x4	
31	reserved_31	ro	rw	0x0	

End RegGroup

End RegGroup

1.22 ROT_KV_RAM

ROT_KV_RAM ▤ 0x00101000, 0x00101004 ... 0x00101FFF

offset		depth	1024	width	32	default	0x0

1.23 ROT_OVERLAY

ROT_OVERLAY ▤ 0x08000000, 0x08000004 ... 0x0FFFFFFF

offset		depth	33554432	width	32	default	0x0