(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2013 (17.10.2013)

WIPO | PCT

(10) International Publication Number
WO 2013/153060 A2

(51) **International Patent Classification**:
G06F 9/44 (2006.01) *G06F 21/74* (2013.01)

(21) **International Application Number**:
PCT/EP2013/057385

(22) **International Filing Date**:
9 April 2013 (09.04.2013)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:

61/621,656	9 April 2012 (09.04.2012)	US
61/663,282	22 June 2012 (22.06.2012)	US

(71) **Applicant**: **SPICER OFF-HIGHWAY BELGIUM N.V.** [BE/BE]; Ten Briele 3, Sint-Michels, B-8200 Brugge (BE).

(72) **Inventors**: **THOMAS, Christophe**; 19 Paradijskouter, B-9031 Drongen (BE). **DUMOULIN, Steven**; 339 Mandellaan, B-8800 Roeselare (BE). **AELVOET, Bjorn**; Wielmakerveld 27, B-9920 Lovendegem (BE).

(74) **Agent**: **GOLKOWSKY, Stefan**; Pfenning, Meinig & Partner GbR, Joachimstaler Strasse 12, 10719 Berlin (DE).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with declaration under Article 17(2)(a); without abstract; title not checked by the International Searching Authority*

WO 2013/153060 A2

(54) **Title**: FUNCTIONAL ARCHITECTURE PATTERN FOR SAFETY APPLICATIONS

(57) **Abstract**:

FUNCTIONAL ARCHITECTURE PATTERN FOR SAFETY APPLICATIONS

RELATED APPLICATION

This application claims the benefit under 35 U.S.C. § 119(e) of U.S. Provisional Patent Application Serial No. 61/621,656, filed April 9, 2012 and U.S. Provisional Patent Application Serial No. 61/663,282, filed June 22, 2012, which are incorporated by reference herein in their entirety.

FIELD OF THE INVENTION

The present invention relates to a functional architecture pattern used in a process to decompose software. More particularly, the present invention relates to a functional architecture pattern used in a process to decompose safety software.

BACKGROUND OF THE INVENTION

Currently, governments and industries are imposing functional safety standards on a variety of entities, for example, the automotive industry. Usually, these functional safety standards are realized by way of safety software that utilizes standard functional architecture patterns. Unfortunately, these functional safety standards, software, and architecture patterns are very rigid and do not give OEMs or integrators enough flexibility to maintain or upgrade the

functionality of the safety-related systems, without incurring considerable added costs to an overall system.

Specifically, functional safety standards require time consuming design and certified validation, where no integration with associated non-safety related applications exists. Hence, expensive hardware and software designs are required because the safety software and standard functional architecture patterns are dedicated. When upgrades are required to a non-safety related application, both the safety application and the associated non-safety related application must be recertified, thereby adding costs and loss of functional system time to an entity.

What is sought is a way to separate the safety related application from the associated non-safety related application, so that when the associated non-safety related application requires maintenance and upgrading, the safety related application does not have to be recertified. Thereby, saving an entity much time and labor costs.

SUMMARY OF THE INVENTION

A process for decomposing software comprises the steps of providing a first software module associated with a first logical unit, providing a second software module associated with a second logical unit, instructing the first software module to implement a first safety goal based on a quality management level, and instructing the second software module to implement a second safety goal based on a safety integrity level, wherein the second software module uses

at least one input and at least one output of the second logical unit to determine if the second safety goal is satisfied. Consequently, the second software module uses a result of the first software module to determine if the first safety goal has been completed, and the second software module uses at least one algorithm to verify an operational status of the first logical unit.

Further objects and advantages of the present invention will be apparent from the following description and appended claims, reference being made to the accompanying drawings forming a part of a specification, wherein like reference characters designate corresponding parts of several views.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a first functional architecture in accordance with present invention;

FIG. 2 is a block diagram of an MPC5668EG processor in accordance with the present invention;

FIG. 3 is a listing of user-level and supervisor-level registers of an e200z6 core of the MPC5668EG processor of Fig. 2;

FIG. 4 is an effective address to real address translation flow diagram in accordance with the present invention;

FIG. 5 is a virtual address and TLB entry compare process diagram in accordance with the present invention;

FIG. 6 is an MMU granting of access permissions diagram in accordance with the present invention;

FIG. 7 is a TLB entry bit fields listing in accordance with the present invention;

FIG. 8 is a block diagram of a second functional architecture in accordance with the present invention; and

FIG. 9 is an architectural structure of memory addresses for SG on an e200z0 core in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

It is to be understood that the invention may assume various alternative orientations and step sequences, except where expressly specified to the contrary. It is also to be understood that the specific devices and processes illustrated in the attached drawings, and described in the following specification are simply exemplary embodiments of the inventive concepts defined in the appended claims. Hence, specific dimensions, directions or other physical characteristics relating to the embodiments disclosed are not to be considered as limiting, unless the claims expressly state otherwise.

Table 1 lists some of the abbreviations used throughout the disclosure.

Table 1: Abbreviations

BSP: Board support packages	RAM: Random Access Memory
ECC: Error correction code	RTOS: Real-time operating system
inet: Internet source	SG: Safety goal(s)
MMU: Memory management unit	SPS: Sharepoint site
MSR: Machine status register	SW-C: Software component(s)

PDF: Portable document format	SW-S/A: Software sensor/ actuator component(s)
PID: Process identifier	TLB: Translation lookaside buffer
QM: Quality Management	uC: Microcontroller

Some of the advantages of the instant invention that will become apparent are the following. Design and validate are only required of a small part of the functional safety related application according to standards known as Safety Integrity Levels (SIL). Design and validation of major parts of the functional safety related application are achieved according to Quality Management (QM) levels. Non-safety related applications (design and validation according to QM level) are integrated with safety related applications. The design for all applications (safety related or not) is achieved with generic hardware/software. Hence, major parts of functional safety related applications are upgraded without re-certification.

Fig. 1 illustrates a first pattern (functional architecture) in accordance with the present invention that uses the following principles. As indicated by the dashed vertical line separating the SW and SG modules, an SIL decomposition is used to make a split-off between safety-related generic reusable software (called the SW module) and safety-related application specific software (called the SG module), wherein the SW and SG modules have their own separate logical unit (uC). Therefore, it is guaranteed that the SW cannot interfere (such as write code, write memory, and/or write data) with the SG.

Also, the SW implements safety goals in a generic functional way according to the QM level. Thereby, the SG implements safety goals in a dedicated application-specific way according to the SIL level on three levels.

Regarding safety-related function monitoring, the SG uses inputs and outputs of its logical unit in order to diagnose if the safety goals are still satisfied. Regarding safety-related application monitoring, the SG uses intermediate results of the SW in order to diagnose if the safety goals are satisfied. Regarding generic application monitoring, the SG uses a question-answer algorithm to check if the SW logical unit is working correctly.

To obtain these results, the following logical entities exist, uC (logical unit) modules are hardware, OS (operating system) module that is developed according to the SIL level, BSP (board support packages) modules that are also developed according to the SIL level, SW (software) module that is developed according to QM level, and SG module that is developed according to the SIL level.

Fig. 2 illustrates the specifics for a global memory manager controller MPC5668EG implemented in both of the preferred embodiments that are respectively illustrated in Figs. 1 and 8. In the supervisor mode/user mode, an e200z6 core (see MASTERS box in upper portion of Fig. 2) defines the following two levels of programming environments. 1) a user level that defines the base user-level instruction set, registers, data types, memory conventions, and the memory and programming models seen by application programmers and 2) a supervisor level that defines supervisor-level resources typically required by

an operating system, i.e., the memory management model, supervisor level registers, and the exception model. However, one skilled in the art recognizes that a different global memory manager controller having the hereinto described elements and functionality of the MPC5668EG device could be used in place thereof.

Fig. 3 shows the e200z6 core register set for the MPC5668EG controller, which indicates registers that are accessible in supervisor mode and those accessible in user mode. The MSR register of the e200z6 core of the MPC5668EG controller, which contains a "problem state" (PR) bit where a "0" indicates that the processor is in a supervisor mode, where any instruction can be executed, and any resource (for example, general purpose GPRs, supervisor purpose SPRs, machine status MSR registers) can be accessed. On the other hand, the "1" bit indicates that the processor is in a user mode, where no privileged instructions can be executed, and no access to any privileged resources is permitted.

The MMU, which is shown in the "Masters" block of the MPC5668EG processor in Fig. 2, is utilized in the present invention to achieve a clear separation between safety-related software and other QM software. To achieve this separation, several architectural choices have been made that impact the requirements for the MPC5668EG processor. The MPC5668EG processor utilizes the supervisor/user mode, MMU, dual RAM, and dual cores (e200z6 and e200z0), wherein the MPC5668EG provides an MMU with 32-bit effective addresses that are translated to 32-bit real addresses using a 41-bit interim

virtual address (see Fig. 4), the MMU may only be available to the e200z6 core and not the e200z0 core, a PID (process identifier) is associated with each effective address (instruction or data), the e200z6 core implements a single PID register (called PID0), where several values for PID0 are defined herebelow.

Further, the translation is done based on TLB entries, specifically, if the PID value in the PID0 matches with a TLB entry in which all of the other criteria are met (see Fig. 6), then that entry will be used for translation. If the translation match was successful, then permission bits are checked. A hit to multiple TLB entries is considered to be a programming error (see Fig. 5), wherein, if this occurs, the TLB generates an invalid address and TLB entries may be corrupted (an exception will not be reported).

Regarding the MMU configuration, the TLB entries need to be defined at initialization time. An example of a TLB entry is shown in the listing of Fig. 7. Also, the MPC5668EG has two SRAM units; one unit with 80 KB on board and a second unit with 512 KB on board, where both units are provided by the ECC. It is preferred that both SRAM units are to be used and that the ECC, along with both e200z6 core and e200z0 cores are to be supported.

An architectural overview of a second pattern is given in Fig. 8 where it is illustrated as a functional architecture that is in accordance with the present invention that uses the following principles. As indicated by the dashed vertical line separating the SW-S/A (sensors/actuators), SW-C (components), and a dRTE (Dana run time environment) module from the SG. The SW-S/A, SW-C, and dRTE modules are considered as developed according to QM standards.

The OS, both BSP's, and SG modules are considered as developed according to SIL standards. Therefore, as in the first functional architectural pattern of Fig. 1, an SIL decomposition is used to make a split-off between a safety-related generic reusable software SW-S/A, SW-C and the safety-related application specific software SG, wherein these SW's and the SG have their own separate logical unit. Therefore, it is guaranteed that the SW's cannot interfere (such as write code, write memory, and/or write data) with the SG.

The second pattern architecture allows the SW-S/A, SW-C, and dRTE modules and OS, both BSP's, and SG modules to reside in the same memory space (code and data space), and on the same uC, without the S/A, SW-C, and dRTE modules being able to corrupt the OS, both BSP's, and SG modules.

For the second pattern, the SG runs on a separate (e200z0) core of the MCP5668EG, where the SW-C only communicates within itself by means of the dRTE. Further, the SW-S/A makes a link between the left BSP and the SW-C. Although the S/A, SW-C, and dRTE are running in supervisor mode (which is discussed in detail herebelow), the S/A, SW-C, and dRTE are shown separately in Fig. 8.

System requirements for the second pattern are to support both the user mode/supervisor mode features, wherein both cores e200z6 and e200z0 are supported, both SRAM units are given ECC support, and the MMU feature is supported. Also, architecture support is given to both the user mode/supervisor mode, both cores e200z6 and e200z0, the 80 KB SRAM and the 512 KB SRAM,

both with ECC. The MMU in the MCP5668EG of the second pattern is integrated with Actia (Automotive Corporation of France) BSP modules.

Fig. 9 illustrates the second pattern from a memory address space point-of-view where the following items are defined. The BSP is the driver code/memory, the OS is supplied by RTOS, SG is the safety-critical code/memory, SW-S/A is the sensor and actuator software components code/memory, SW-C module is the application software components.

OS interrupts run in supervisor mode, BSP interrupts run in supervisor mode, and the SW-S/A runs in supervisor mode, where the SW-S/A is the only software component(s) that may access BSP. Also, the SW-S/A may read and write the shared memory.

The SW-C runs in user mode but may not access BSP, nor read and write the shared memory. The SG runs in supervisor mode and may access BSP and may read the shared memory.

The second architectural concept of Fig. 9, taken from a memory address space point of view, has access rights to various memory address spaces, for the different modules. As shown down the left side of Fig. 9, the rights are defined as - SR: Supervisor read, SW: Supervisor write, SX: Supervisor execute, UR: User read, UW: User write, UX: User execute. These access rights for the different modules in Fig. 9, are as follows.

The following memory address spaces have SR,SX (shown in normal font orientation) – lower BSP and OS. The following memory address spaces have *SR,SX* (shown in *italicized* font orientation) – lower BSP and SW-S/A. The

following memory address space has **UR,UX** (shown in bold font orientation) – lower SW-C. The following memory address spaces have SR,SW (shown in normal font orientation) – upper BSP, SHARED, OS, and both HARDWARE REGISTERS. The following memory address spaces have *SR,SW* (shown in *italicized* font orientation) – upper BSP, SW-S/A, SHARED, and both HARDWARE REGISTERS. The following memory address space has **UR,UW** (shown in bold font orientation) – upper SW-C and SHARED. SG has rights on the entire memory address space on the e200z0 core, with no MMU (shown in a dashed box on the right of the memory address space).

The memory address space of Fig. 9 is configured as a static configuration, and is based on a linker output. This provides the user flexibility to define the regions, rights and the corresponding sizes.

Only BSP configures the MMU, such that when a miss is detected by the MMU, the BSP makes sure that the controller resets. The OS will configure the user and supervisor mode for the SW-S/A and SW-C and the OS will set the corresponding PID.

The basic safety strategy plans are: 1) ASIL (ISO 26262) decomposition principle is used to make a split-off between safety-related generic re-usable software (while following the QM process of Dana dFLAME software) and safety-related application specific software (implementing specifically the safety goals for a specific application while following an SIL process in dFLAME. 2) The safety-related generic re-usable software is mapped onto the SW-C and SW-S/A modules. 3) The safety-related application specific software is mapped onto the

SG module. 4) The SW-C and SW-S/A cannot corrupt the SG code space and memory space. 5) The ECC is used for all entities, for both code space and memory space. 6) The SW-C and SW-S/A run on the e200z6 core. 7) The SG runs on the e200z0 core. 8) The SW-C and SW-S/A use the 512 KB SRAM unit. 9) The SG uses the 80 KB SRAM unit. 10) Both cores monitor each other periodically. 11) BSP is developed according to SIL. The monitoring of both cores is also known as "controller monitoring" (for example, E-Gas monitoring concept for engine management systems of gasoline and diesel engines).

This type of monitoring is function (application) independent, by way of question-answer algorithms.

Consequently, SG implements the safety goals in an application-specific way where two levels are defined: 1) Outputs are directly correlated to inputs and checked for safe behavior. 2) Function monitoring where Intermediate results in the SW-C and SW-S/A are checked for safe behavior.

Safety-related data that comes over CAN (e.g. engine speed) must be both available for the SW-S/A and SG. The callback of the CAN is to be developed according to SIL and specifically route safety related data to SG.

BSP is to be able to support both calls from SW-S/A and SG.

Consequently, the advantages of the instant invention are that the design and validation are only required of a small part of the functional safety related application according to SIL. The major parts of the functional safety related application are achieved according to QM levels and non-safety related applications are integrated with safety related applications. The design for all

applications (safety related or not) is achieved by way of generic hardware/software. Hence, major parts of functional safety related applications are upgraded without re-certification. Thereby, saving an entity much time and labor costs.

In accordance with the provisions of the patent statutes, the present invention has been described in what is considered to represent its preferred embodiments. However, it should be noted that the invention can be practiced otherwise than as specifically illustrated and described without departing from its spirit or scope.

CLAIMS:

1. A functional architecture pattern comprising:

a safety-related generic reusable sensor/actuator software module SW-S/A, a safety-related generic reusable component software module SW-C, a run time environment module dRTE, an operating system module OS, a first board support package BSP, and a first microcomputer uC;

a safety-related application specific software module SG, a second board support package BSP, and a second microcomputer uC that are logically separated from the sensor/actuator software module SW-S/A, component software module SW-C, run time environment module dRTE, operating system module OS, the first board support package BSP, and the first microcomputer;

wherein the sensor/actuator software module SW-S/A, the component software module SW-C, and the run time environment module dRTE are developed according to QM standards; and

the OS, both first and second BSPs, and the SG modules are developed according to SIL standards so as to make a logical split-off between the SW-S/A, SW-C software and the safety-related application specific software SG.

2. The functional architecture pattern of claim 1, wherein the sensor/actuator software module SW-S/A and the component software module SW-C comprise a logical unit separate from the safety-related application specific software SG so as to have separate write code, write memory, and/or write data from the safety-related application specific software SG.

3. The functional architecture pattern of claim 1, wherein the SW-S/A, SW-C, dRTE modules, OS, both BSP's, and the SG modules reside in the same memory code and space, and on the same microcomputer uC, without the OS, both BSP's, and SG modules being corrupted by the SW-S/A, SW-C, and dRTE modules.

4. The functional architecture pattern of claim 1, wherein the SG runs on a separate core (e200z0) of the microcomputer uC that comprises an MCP5668EG and the SW-C is only in communications within itself by way of the dRTE.

5. The functional architecture pattern of claim 1, wherein the SW-S/A is linked between the first BSP and the SW-C.

6. The functional architecture pattern of claim 1, wherein the SW-S/A, SW-C, and dRTE operate in supervisor mode.

7. The functional architecture pattern of claim 1, wherein both user mode and supervisor mode features, both cores e200z6 and e200z0, both SRAM units having ECC, and the MMU are supported.

8. The functional architecture pattern of claim 1, wherein both user mode/supervisor modes, both cores e200z6 and e200z0, an 80 KB SRAM, a 512 KB SRAM, and an MMU in an MCP5668EG that are integrated with BSP modules are architecturally supported.

9. The functional architecture pattern of claim 1, wherein the OS is supplied by RTOS, the OS and BSP interrupts in supervisor mode, the SW-S/A is operated in supervisor mode and only software component having BSP access, and the SW-S/A has read and write capability with shared memory.

10. The functional architecture pattern of claim 1, wherein when the SW-C is operated in user mode, the BSP is not accessible by SW-C, the SW-C is not read and write capable of shared memory, and the SG is operated in supervisor mode and is BSP accessible, and is readable of shared memory.

11. The functional architecture pattern of claim 1, wherein the SG is determinable of a correct working of the S/A, SW-C logical units by way of a question-answer algorithm.

12. A functional architecture pattern comprising:

a safety-related generic reusable software module SW, an operating system module OS, a first board support package BSP, and a first microcomputer uC;

a safety-related application specific software module SG, a second board support package BSP, and a second microcomputer uC that are logically separated from the safety-related generic reusable software module SW, the operating system module OS, the first board support package BSP, and the first microcomputer uC;

wherein the safety-related generic reusable software module SW is developed according to QM standards; and

the OS, both first and second BSPs, and the SG modules are developed according to SIL standards so as to make a logical split-off on three levels between the SW software and the safety-related application specific software SG.

13. The functional architecture pattern of claim 12, wherein inputs and outputs of the SG are monitored by a logical unit of SG to diagnose safety goal satisfaction.

14. The functional architecture pattern of claim 12, wherein the SG is capable of intermediate results of the SW for diagnosed safety goals satisfaction.

15. The functional architecture pattern of claim 12, wherein the SG has a question-answer algorithm determinable of a correct operation of an SW logical unit.

16. A process for decomposing software, comprising the steps of:

providing a first software module SW associated with a first logical unit;

providing a second software module SG associated with a second logical unit;

instructing the first software module SW to implement a first safety goal based on a quality management QM level; and

instructing the second software module SG to implement a second safety goal based on a safety integrity level SIL, wherein the second software module SG uses at least one input and at least one output of the second logical unit to determine if the second safety goal is satisfied, the second software module SG uses a result of the first software module SW to determine if the first safety goal has been completed, and the second software module SG uses at least one algorithm to verify an operational status of the first logical unit.

17. The process for decomposing software of claim 16, further comprising the steps of:

providing a third software module SW-C that is a safety-related generic reusable component software module associated with the first logical unit,

wherein the first software module SW comprises a safety-related generic reusable sensor/actuator software module SW-S/A;

instructing the first software module SW-S/A and third software module SW-C to implement a first safety goal based on a quality management QM level; and

instructing the second software module SG to implement a second safety goal based on a safety integrity level SIL, wherein the second software module SG uses at least one input and at least one output of the second logical unit to determine if the second safety goal is satisfied, the second software module SG uses a result of the first and third software modules SW-S/A, SW-C to determine if the first safety goal has been completed, and the second software module SG uses at least one algorithm to verify an operational status of the first logical unit.

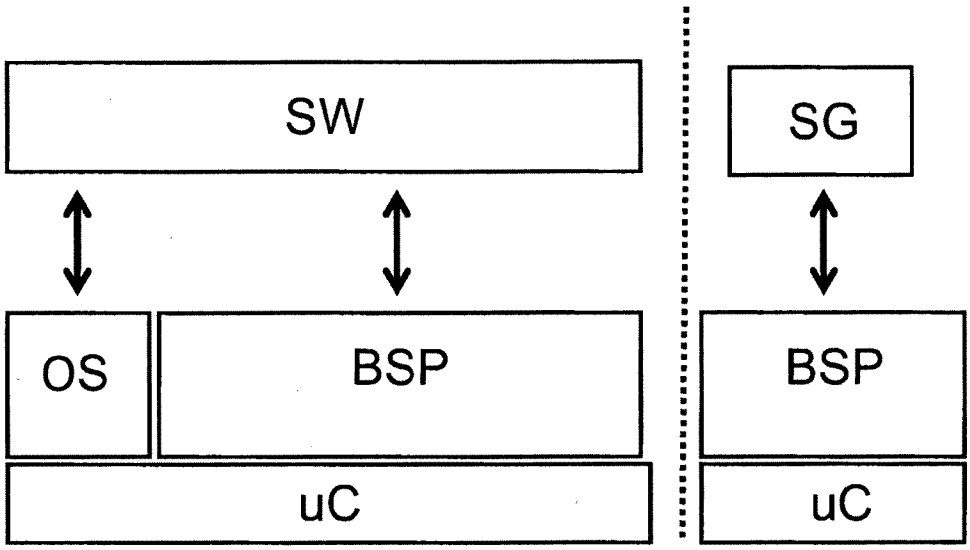


Fig. 1

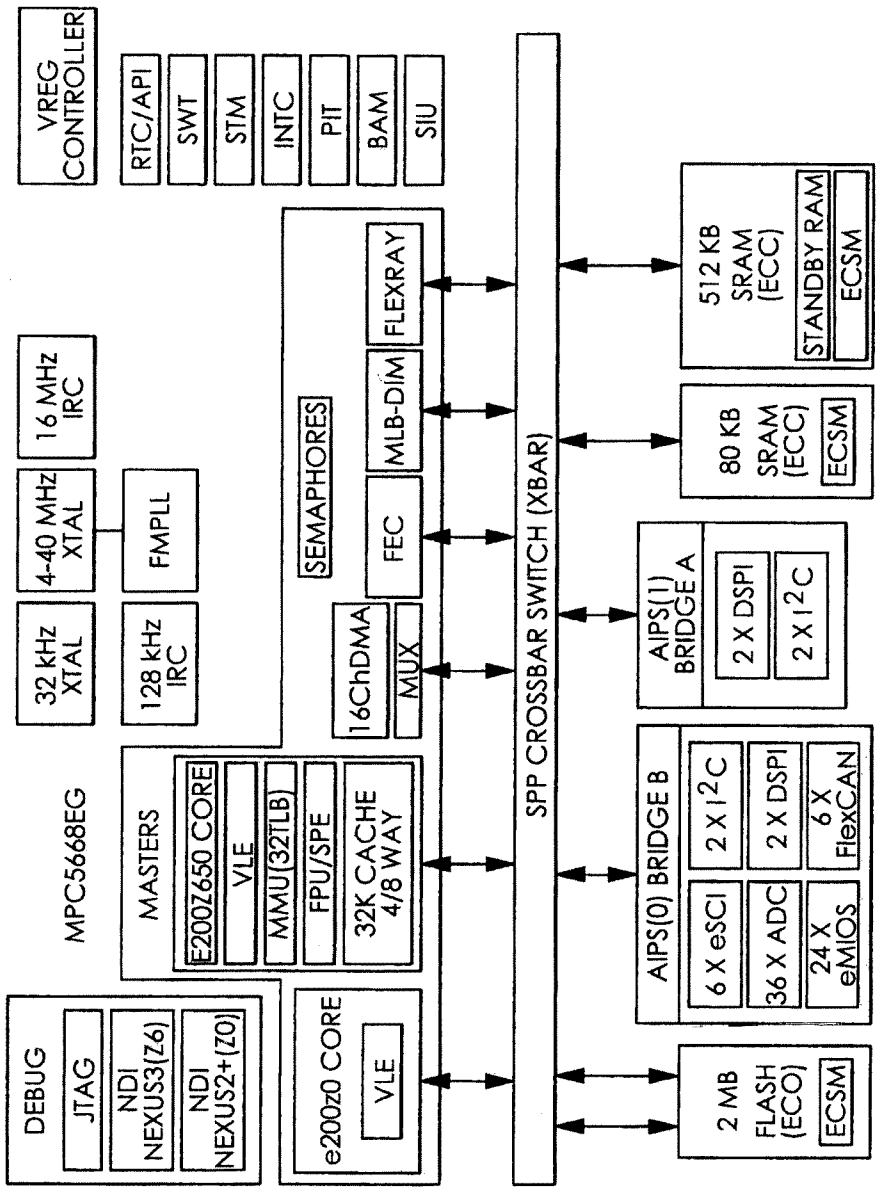
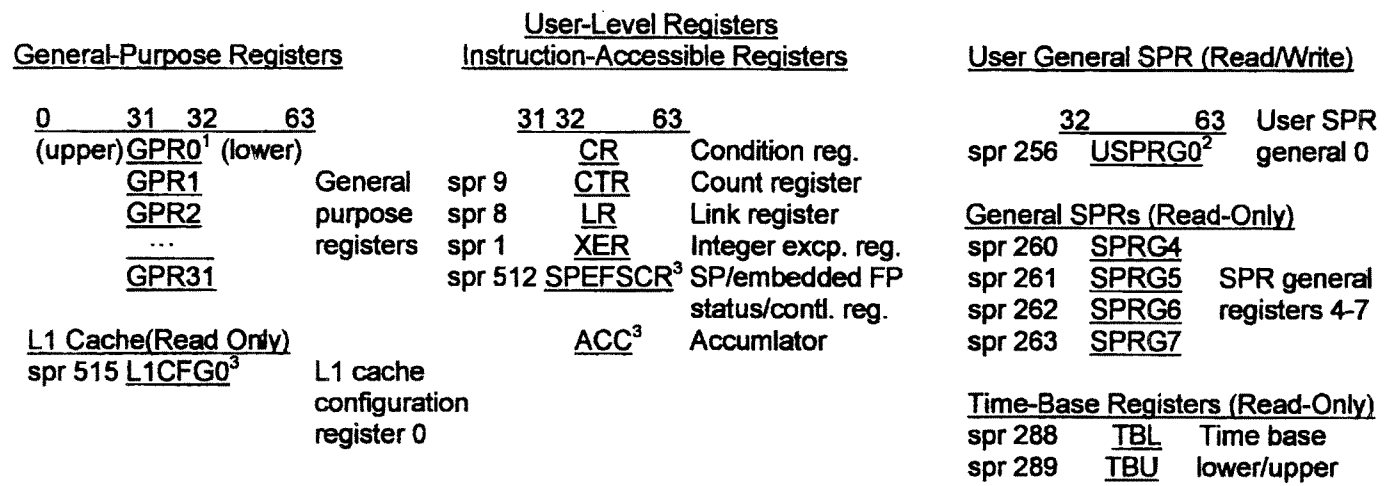

VREG CONTROLLER
RTC/API
SWT
STM
INTC
PIT
BAM
SIU
16 MHz IRC
4-40 MHz XTAL
FMPLL
32 kHz XTAL
128 kHz IRC
SEMAPHORES
MPC5668EG
MASTERS
E200Z650 CORE
VLE
MMU(32TLB)
FPU/SPE
32K CACHE 4/8 WAY
16ChDMA
MUX
FEC
MLB-DIM
FLEXRAY
DEBUG
JTAG
NDI NEXUS3(Z6)
NDI NEXUS2+(Z0)
e200z0 CORE
VLE
SPP CROSSBAR SWITCH (XBAR)
AIPS(1) BRIDGE A
2 X DSPI
2 X I2C
AIPS(0) BRIDGE B
6 X eSCI
36 X ADC
24 X eMIOS
2 X I2C
2 X DSPI
6 X FlexCAN
512 KB SRAM (ECC)
STANDBY RAM
ECSM
80 KB SRAM (ECC)
ECSM
2 MB FLASH (ECO)
ECSM

FIG. 2



[1]The 64-bit GPR registers are accessed by the SPE as separate 32 bit registers by SPE instructions. Only SPE vector instructions can access the upper-word.
[2]USPRG0 is a separate physical register from SPRG0.
[3]EIS-specific registers; not part of the Book E architecture.

FIG. 3

Supervisor-Level Registers

Interrupt Registers

	32 63	
spr 63	IVPR	Intrupt. vect. prefix reg.
spr 26	SRR0	Save/restore
spr 27	SRR1	registers 0/1
spr 58	CSRR0	Critical SRR 0/1
spr 59	CSRR1	
spr 574	DSRR0[3]	Debug interrupt
spr 575	DSRR1[3]	SRR0/1
spr 62	ESR	Except. syndrome register
spr 572	MCSR[3]	Machine check syndrome register
spr 61	DEAR	Data exception address register

	32 63	
spr 400	IVOR0	
spr 401	IVOR1	Intrupt. vect. offset registers 0-15[4]
...		
spr 415	IVOR15	
spr 528	IVOR32[3]	
spr 529	IVOR33[3]	Intrupt. vect. offset registers 32-34
spr 530	IVOR34[3]	

MMU Control and Status (Read/Write)

spr 1012	MMUCSR0[3]	MMU contl. & status reg. 0
spr 624	MAS0[3]	
spr 625	MAS1[3]	
spr 626	MAS2[3]	MMU asst. reg. 0-4 and 6
spr 627	MAS3[3]	
spr 628	MAS4[3]	
spr 630	MAS6[3]	
spr 48	PIDO	Process ID register 0

Configuration Registers

	32 63	
	MSR	Mach. state reg.
spr 1023	SVR[3]	System version register
spr 286	PIR	Processor ID reg.
spr 287	PVR	Processor version register

Timer/Decrementer Registers

spr 22	DEC	Decrementer
spr 54	DECAR	Decrementer auto-reload reg.
spr 284	TBL	Time base
spr 285	TBU	lower/upper
spr 340	TCR	Time control reg.
spr 336	TSR	Time status reg.

[4]IVOR9 (handles auxiliary processor unavailable interrupt) is defined by the EIS but not supported by the e200z6.

FIG. 3 (Continuing)

Supervisor-Level Registers

Interrupt Registers

Debug Registers

32	63	
spr 308	DBCR0	
spr 309	DBCR1	Debug control
spr 310	DBCR2	register 0-3
spr 561	DBCR3	
spr 304	DBSR	Debug status reg.
spr 562	DBCNT[5]	Debug count reg.
spr 312	IAC1	
spr 313	IAC2	Instruction address
spr 314	IAC3	compare
spr 315	IAC4	register 1-4
spr 316	DAC1	Data address
spr 317	DAC2	compare registers 1 and 2

MMU Control and Status (Read Only)

32	63	
spr 1015	MMUCFG[3]	MMU config.
spr 688	TLB0CFG[3]	TLB config.
spr 689	TLB1CFG[3]	0/1

L1 Cache (Read/Write)

spr 1010	L1CSR0[3]	L1 cache contl./ status reg. 0
spr 1018	L1FINV0[5]	L1 cache flush/invalid. contl. register 0

Configuration Registers

Miscellaneous Registers

32	63	
spr 1008	HID0[3]	Hardware
spr 1009	HID1[3]	Implementation dependent 0-1
spr 1013	BUCSR[5]	Branch control and status reg.
spr 272-279	SPRG0-7	Gen. SPRs 0-7

Context Control (Read/Write)

spr 560	CTXCR[5]	Context control register
spr 568	ALTCTXCR[5]	Altern. context control register

[5]e200z6 –specific registers

FIG. 3 (Continuing)

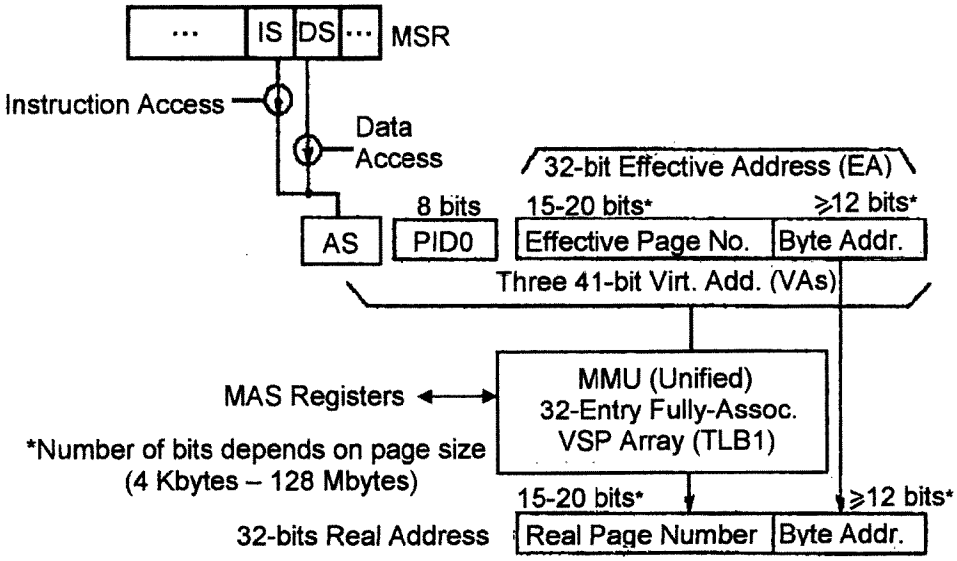

...
IS
DS
...
MSR
Instruction Access
Data Access
32-bit Effective Address (EA)
8 bits
15-20 bits*
≥12 bits*
AS
PID0
Effective Page No.
Byte Addr.
Three 41-bit Virt. Add. (VAs)
MAS Registers
MMU (Unified)
32-Entry Fully-Assoc.
VSP Array (TLB1)
*Number of bits depends on page size
(4 Kbytes – 128 Mbytes)
15-20 bits*
≥12 bits*
32-bits Real Address
Real Page Number
Byte Addr.

FIG. 4

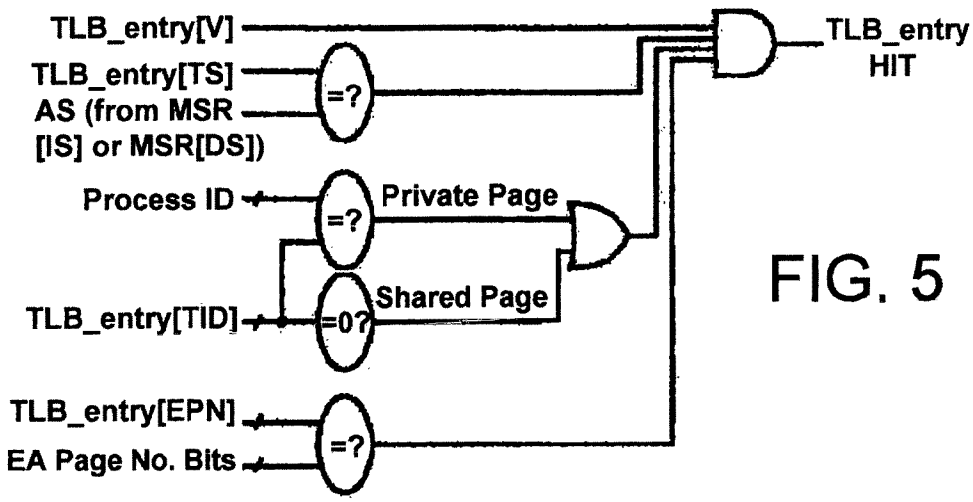

TLB_entry[V]
TLB_entry[TS]
AS (from MSR
[IS] or MSR[DS])
=?
TLB_entry
HIT
Process ID
=?
Private Page
Shared Page
TLB_entry[TID]
=0?
TLB_entry[EPN]
EA Page No. Bits
=?

FIG. 5

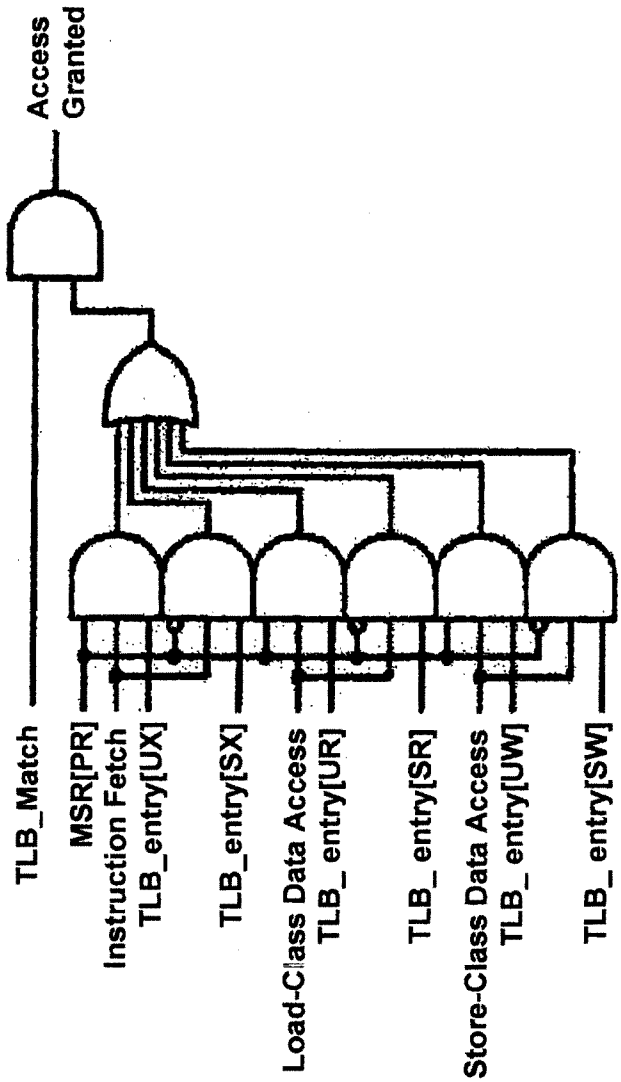

Access Granted
TLB_Match
MSR[PR]
Instruction Fetch
TLB_entry[UX]
TLB_entry[SX]
Load-Class Data Access
TLB_ entry[UR]
TLB_ entry[SR]
Store-Class Data Access
TLB_ entry[UW]
TLB_ entry[SW]

FIG. 6

Field	Description
V	Valid bit for entry
TS	Translation address space (compared with AS bit of the current access)
TID [0-7]	Translation ID (compared with PID0 or TIDZ (all zeros))
EPN [0-19]	Translation ID (compared with PID0 or TIDZ (all zeros))
RPN [0-19]	Real page number (translated address)
	Encoded page size
	0000 Reserved
	0001 4 Kbytes
	0010 16 Kbytes
	0011 64 Kbytes
SIZE [0-3]	0100 256 Kbytes
	0101 1 Mbytes
	0110 4 Mbytes
	0111 16 Mbytes
	1000 64 Mbytes
	1001 256 Mbytes
	All others --- reserved
PERMIS [0-5]	Supervisor execute, write, and read permission bits, and user execute, write, and read permission bits
WIMGE	Memory / cache attributes (write-through, cache-inhibit, memory Coherence required, guarded, endian)
U0-U3	User attribute bitsNused only by software
IPROT	Invalidation protection

FIG. 7

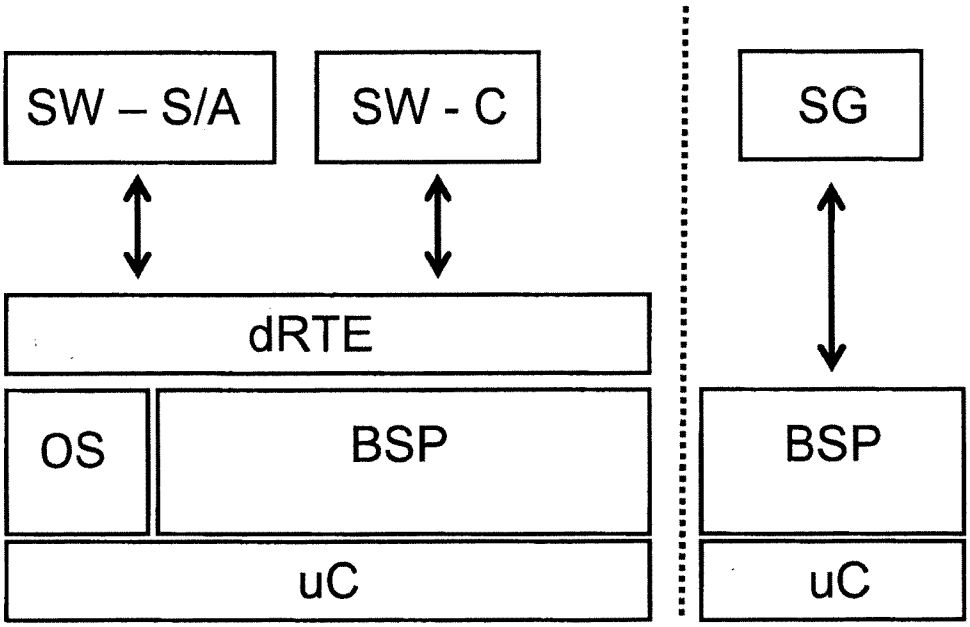


Fig. 8

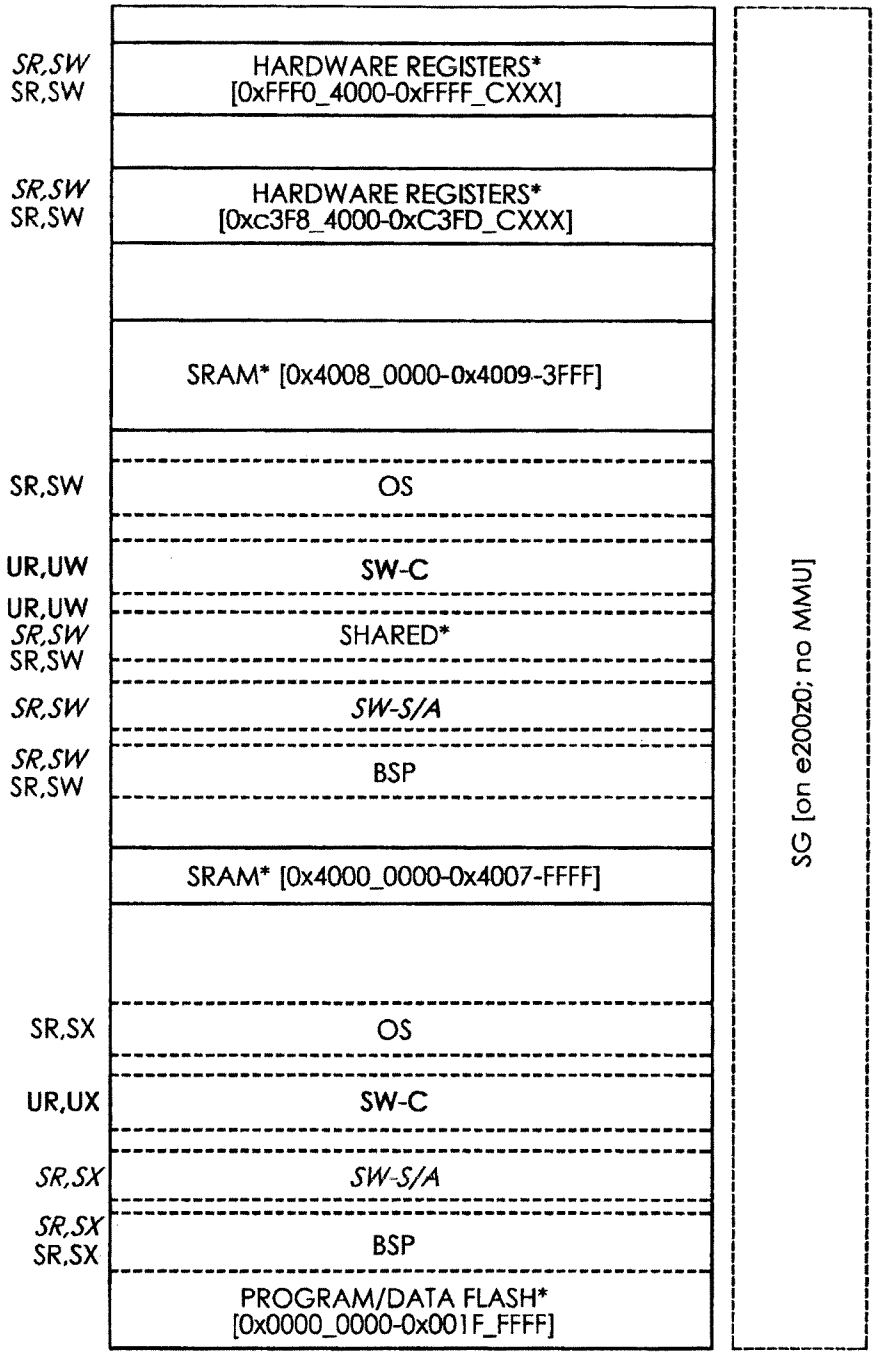

SR,SW
SR,SW
HARDWARE REGISTERS*
[0xFFF0_4000-0xFFFF_CXXX]
SR,SW
SR,SW
HARDWARE REGISTERS*
[0xc3F8_4000-0xC3FD_CXXX]
SRAM* [0x4008_0000-0x4009-3FFF]
SR,SW
OS
UR,UW
SW-C
UR,UW
SR,SW
SR,SW
SHARED*
SR,SW
SW-S/A
SR,SW
SR,SW
BSP
SRAM* [0x4000_0000-0x4007-FFFF]
SR,SX
OS
UR,UX
SW-C
SR,SX
SW-S/A
SR,SX
SR,SX
BSP
PROGRAM/DATA FLASH*
[0x0000_0000-0x001F_FFFF]
SG [on e200z0; no MMU]

FIG. 9

PATENT COOPERATION TREATY

PCT

DECLARATION OF NON-ESTABLISHMENT OF INTERNATIONAL SEARCH REPORT

(PCT Article 17(2)(a), Rules 13*ter*.1(c) and Rule 39)

Applicant's or agent's file reference	IMPORTANT DECLARATION	Date of mailing *(day/month/year)*
137PCT0651		9 July 2013 (09-07-2013)
International application No.	International filing date *(day/month/year)*	(Earliest) Priority date *(day/month/year)*
PCT/EP2013/057385	9 April 2013 (09-04-2013)	9 April 2012 (09-04-2012)
International Patent Classification (IPC) or both national classification and IPC		
G06F8/10, G06F21/74		
Applicant		
SPICER OFF-HIGHWAY BELGIUM N.V.		

This International Searching Authority hereby declares, according to Article 17(2)(a), that **no international search report will be established** on the international application for the reasons indicated below

1. [X] The subject matter of the international application relates to:
   - a. [ ] scientific theories
   - b. [ ] mathematical theories
   - c. [ ] plant varieties
   - d. [ ] animal varieties
   - e. [ ] essentially biological processes for the production of plants and animals, other than microbiological processes and the products of such processes
   - f. [ ] schemes, rules or methods of doing business
   - g. [X] schemes, rules or methods of performing purely mental acts
   - h. [ ] schemes, rules or methods of playing games
   - i. [ ] methods for treatment of the human body by surgery or therapy
   - j. [ ] methods for treatment of the animal body by surgery or therapy
   - k. [ ] diagnostic methods practised on the human or animal body
   - l. [ ] mere presentations of information
   - m. [ ] computer programs for which this International Searching Authority is not equipped to search prior art
2. [X] The failure of the following parts of the international application to comply with prescribed requirements prevents a meaningful search from being carried out:
   [X] the description [X] the claims [X] the drawings
3. [ ] A meaningful search could not be carried out without the sequence listing; the applicant did not, within the prescribed time limit:
   - [ ] furnish a sequence listing on paper complying with the standard provided for in Annex C of the Administrative Instructions, and such listing was not available to the International Searching Authority in a form and manner acceptable to it.
   - [ ] furnish a sequence listing in electronic form complying with the standard provided for in Annex C of the Administrative Instructions, and such listing was not available to the International Searching Authority in a form and manner acceptable to it.
   - [ ] pay the required late furnishing fee for the furnishing of a sequence listing in response to an invitation under Rule 13*ter*.1(a) or (b).
4. Further comments:

Name and mailing address of the International Searching Authority

European Patent Office, P.B. 5818 Patentlaan 2
NL-2280 HV Rijswijk
Tel. (+31-70) 340-2040
Fax: (+31-70) 340-3016

Authorized officer

GARRY, Cora
Tel: +49 (0)89 2399-5771

FURTHER INFORMATION CONTINUED FROM PCT/ISA/ 203

The subject-matter claimed relates to a functional architecture pattern (claims 1-15) and to a process for decomposing software (claims 16 and 17).
The wording of the claims does not go beyond well-known high-level commonplace concepts for software and hardware architecture, such as different kinds of software modules, software development principles, logical separation between software and hardware and architecture of microcomputers. The subject-matter claimed does not lead to any technical effect nor does it solve any technical problem. In particular, a "functional architecture pattern" as claimed is merely an abstract construct comprising of abstract rules which do not lead to any technical effect. Similarly, the "process for decomposing software" as claimed is merely a sequence of steps to be carried out by a programmer, without any technical effect being apparent.
Furthermore, many of the claimed features are formulated as a result to be achieved, without any indication in the claims or indeed in the entire application how, in terms of technical features, the desired result is to be achieved. In particular, the logical separation/split-off, the avoidance of corruption, the determining of a correct working, the diagnosing of safety goal satisfaction are all unclear because of this formulation.
For these reasons, and due to the very generic drafting and the extreme high abstraction level of the formulation of the claims, no meaningful search is possible, Article 17(2)(a) PCT and the PCT guidelines, PCT/GL/ISPE, Part III, chapter 9.04.

The applicant's attention is drawn to the fact that claims relating to inventions in respect of which no international search report has been established need not be the subject of an international preliminary examination (Rule 66.1(e) PCT). The applicant is advised that the EPO policy when acting as an International Preliminary Examining Authority is normally not to carry out a preliminary examination on matter which has not been searched. This is the case irrespective of whether or not the claims are amended following receipt of the search report or during any Chapter II procedure. If the application proceeds into the regional phase before the EPO, the applicant is reminded that a search may be carried out during examination before the EPO (see EPO Guidelines C-IV, 7.2), should the problems which led to the Article 17(2) declaration be overcome.