Design and Test of Microcontroller with CAN 2.0B embedded for Space Applications

Egidio Pescari - Aurelia Microelettronica S.p.A.

Andrea S. Brogna, Roberto Saletti - Università degli Studi di Pisa

Abstract – This paper presents a general purpose CPU with CAN bus controller designed to work in space and hostile environments where the effects of ionizing particles are relevant. Special design techniques and redundancy protect the internal registers and the memory cells - both internal and external - from Single Event Upset, with minimal impact on the performances and without using a radiation hardness technologic process for the chip. Moreover, an embedded In Circuit Emulator helps to develop and debug the software applications.

Introduction

Safety and simplicity of CAN protocol make it very suitable for critical applications in harsh environments, such as automotive. These interesting features are very appreciated in space applications where solutions based on largely used protocol and systems are not expansive and require less efforts in their implementation.

Some commercial devices already integrate a CPU and CAN controller, but a slow control for a satellite on board equipment requires a special care to handle Single Event Upsets (SEU) due to ionizing particles which might alter the state of a memory cell or registers and cause a system failure.

Moreover, space agencies and companies involved in aerospace systems require that each flight part meets all test specifications [1] and qualification QML-Q or QML-V.

SPECIES (SEU Protected CAN controller with In Circuit Emulator for Space) is a system on chip with a CPU based on 8051 architecture which integrates a CAN controller: it is designed to minimize the effects of SEU because of its internal architecture, and with a radiation hardness process, the chip can resist also to latch-up and total ionizing dose effects: in this case it would be qualified to flight into space.

It is an evolution of CASA2 chip [2], a stand alone CAN controller already installed in the Automated Transfer Vehicle (ATV), but the embedded CAN controller in SPECIES has improved features and enhancements such as a transmitter and receiver programmable mailboxes with individual acceptance filter.

Moreover, an In Circuit Emulator (ICE) is available to help the development and the debug of software in high-level languages.

The Architecture

SPECIES has a CPU core based on 8051 which is an almost standard architecture in industrial applications. The system includes also the CAN controller, an Universal Asynchronous Receiver Transmitter (UART) which is

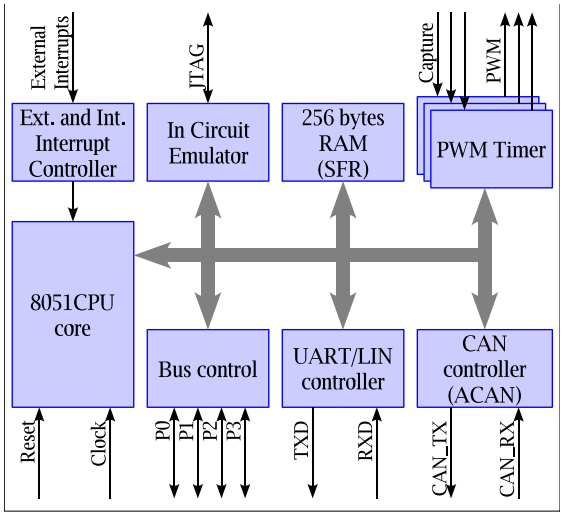

Figure 1 - The SPECIES system on chip architecture

programmable/configurable as Local Interconnect Network (LIN) controller, 3 Timers and an In Circuit Emulator (ICE) to develop and debug software based applications. Figure 1 shows the block diagram with interconnections of the SPECIES.

All the peripherals can signal special condition with an interrupt flag. The Interrupt Control Unit handles the interrupt sources (including the external ones) and assigns them a priority: a programmable register allows to mask the flags so that the service routines can optimize the latency at system level.

The timers work as PWMs (Pulse Width Modulator) or as counters; in the first case they generate a periodic square waveform with programmable period and duty cycle, in the second case the free running counter can be compared to a predefined value (compare mode) or latched by external event (capture mode).

The programmable UART and LIN controller can transmit and receive using UART LIN protocol in full duplex mode and, as a LIN controller, it can be a master or a slave. Internal clock or external source provides the timing.

The LIN is a serial communication protocol which efficiently supports the control of mechatronic nodes in a distributed automotive applications.

The CAN controller

This module implements a fully compliant CAN 2.0B protocol including acceptance filtering function and some additional features:

- programmable data rates;
- power down in sleep mode;
- up to 16 transmit and receive mailbox;
- up to 16 individual acceptance filtering;
- interrupt event sources for:
  - telegram sent successfully;
  - telegram received successfully;
  - receive buffer overflow;
  - bus off condition;
  - error passive condition;

The embedded CAN controller (ACAN) performs all serial communication

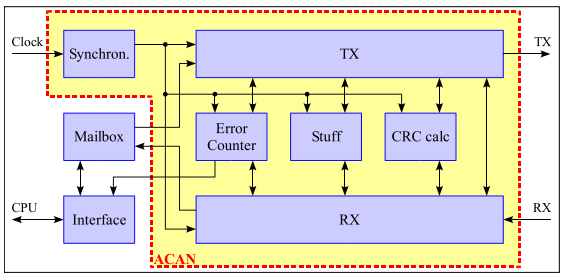

Figure 2 - The CAN Controller (ACAN) and its interface to the CPU

according to the CAN protocol which uses multi-master bus configuration to transfer data packets between nodes on network.

Figure 2 shows the internal structure of the ACAN block which is the properly CAN controller and the interface parts to integrate and extend its functionality.

The CAN controller supports both standard and extended messages frame format as described in 2.0B specifications with 29 bit identifier and it can filter also a 29 bit receiving message.

Additionally there are 16 configurable mailboxes in transmitting or receiving state, mapped on two 128x8 internal dual port synchronous RAMs.

The filter function applies to each mailbox which consists of a register to store the 29 bit identifier, the data to be transmitted or received and two programmable registers to store the status: if a mailbox is in the receive state, the identifier allows to filter the incoming message with a global mask (stored in four 8-bytes registers) that implements a "*don't care*" condition.

The incoming message is accepted and stored in the mailbox only if its identifier matches the programmed identifier but if it matches more than one identifier, the message will be stored in the lowest numbered mailbox programmed in receive state.

An embedded Built In Self Test (BIST) circuit checks the functionality and the status of the internal RAM.

According to the CAN 2.0B specifications, the Error Counter block manages errors, and the CRC Calc block calculates the Cyclical Redundancy Check (CRC) to append to the transmitting data or check the incoming data to detect errors.

CAN 2.0B bit timing function

The CAN protocol allows the user to program the bit rate and the sample point of the bit because of the relationship among bit timing parameters, the physical bus propagation delays and the oscillator tolerances throughout the system [3].

SPECIES has a full control to optimize the bit timing for given physical system attributes and network parameters.

Figure 3 shows a typical bit timing situation. The CAN bit timing is a sequence of non-overlapping segments which are made of integer units called Time Quanta. The period of the system clock Tscl defines the Time Quantum length. A programmable Baud Rate Prescaler (BRP) can set up the Tscl period to 1, 2, 4 or 8 times the system clock period.

The Synchronization Segment (Sync) is that part of the bit time where edges of the CAN bus level are expected to occur.

The Phase Segment #1 and Phase Segment #2 surround the Sample Point. The Resynchronization Jump Width (RSJ) defines how far a resynchronization may move the Sample Point inside the limits

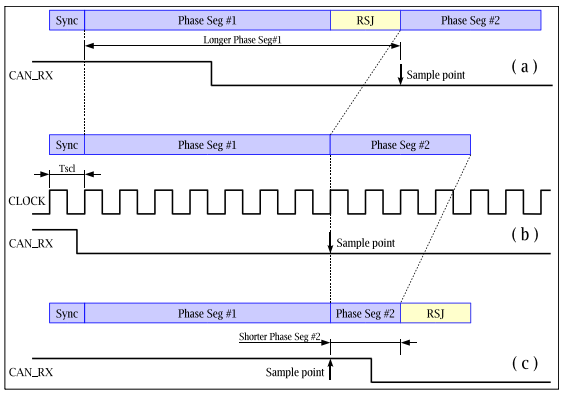

Figure 3 - Bit timing function as implemented in SPECIES. An edge is synchronous if it occurs inside of Sync Segment as in (b), otherwise the distance between edge and the end of Sync Segment is the edge phase error, measured in time quanta. In (a) the edge occurs after Sync Segment and the phase error is negative. In (c) it occurs before Sync Segment and it is positive.

defined by the Phase Buffer Segments to compensate for edge phase errors.

The Phase Segment #1 and Phase Segment #2 and Resynchronization Jump are used to compensate the oscillator tolerance. Each Phase Segment may be lengthened or shortened by synchronization block.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the actual time quantum bus level is dominant.

An edge is synchronous if it occurs inside of Sync Segment, otherwise the distance between edge and the end of Sync Segment is the edge phase error, measured in time quanta. If the edge occurs after Sync Segment, the phase error is negative (Figure 3 (a)), else it is positive (Figure 3 (c)).

Phase Segment #1 is programmable from 2 to 16 Tscl and Phase Segment #2 is programmable from 1 to 8 Tscl

Re-Synchronisation Jump (RSJ) width is used to compensate phase shifts between oscillator frequency of different CAN nodes on network. This value is programmable from 1 to 4 Tscl

The Bit rate of message on bus is calculated as:

$$bit\ rate = \frac{Fclock}{BRP\ (PH\ SEG\#1 + PH\ SEG\#2 + 1)}$$

Transmission of messages

Before sending a message, the ACAN module is initialised with proper values for registers to control interrupt generator configuration, data rate, bit timing configuration, message length and mailbox configuration; then to transmit a message on the bus, the user must program the configuration registers of CAN controller. After this, the CPU sets the bit to update the status of the message

buffer and the bit to start transmission. At this point, the message buffer is sent to bus line. The ACAN begins the transmission from data stored in the lowest numbered mailbox that is programmed in transmission mode. ACAN sends as many message as are the number of mailbox programmed in transmission, starting from the lowest and ending to mailbox number 16.

The status register of ACAN module which contains error condition, bus-off condition, transmission active or transmission OK information is always readable by CPU.

Incoming message

The incoming message passes through the global mask and it is checked with the identifier of the first mailbox in receive state. If the message is accepted, the data will be stored in the first mailbox and the status of receiver mailbox will be updated with the information of received message: length of message, extended or standard frame message, reception OK or overrun condition. The incoming message is stored in the mailbox at the end of END of FRAME field (7 recessive bits).

If the incoming message is a remote frame request from another CAN node, the incoming remote request will not be stored in the mailboxes.

Trigger match function

Trigger match function is a feature implemented to generate HATRIG signal (a pulse for 13 clock cycle) if the identifier of incoming message matches a user-programmed identifier value. This function is not related to filtering of message and interrupt generation. The HATRIG signal, together with CAN node regularly generating the appropriate frame, could be used to force the bus switching in a system with two physical bus used as nominal and redundant. This feature is very interesting in applications where redundancy system are required.

The In Circuit Emulator

This block is in charge of the program upload/downloading and hardware support for the C/ASSEMBLY program debugging because it accesses to the 8051 Special Function Registers (SFR) and the external program/data memory [4].

It communicates with the host environment using a JTAG interface which internally serializes and de-serializes operations on the JTAG stream. It contains the Test Access Port (TAP) controller and a simple interface used to read and write data. The internal state of this block depends only from external JTAG signals because it is asynchronously respect to the 8051 clock.

The ICE controller synchronizes the signals from the JTAG TAP with the CPU core and the BIST logic block.

The program flow control and program code swapping are also implemented in this module. It stretches the system clock and injects data and address on the bus between the 8051 micro and the SFR block. Moreover, it is also possible to inspect the SFR.

More in detail the interaction with the 8051 environment happens in 3 distinct operating modes:

Single Cycle : which means a step by step tracing analysis. The 8051 executes a single instruction and it is immediately stopped when it starts to fetch the new one.

Single Cycle with Enforced Code: The 8051 is forced to execute a specific instruction code supplied via JTAG. The combined execution of the MOVX code with the direct access to the SFR registers makes possible the uploading/ downloading of the external memory with program and data.

Break point: It is possible to insert up to 4 hardware breakpoints in a program during debugging phases. The break of the program execution in a specific point is achieved loading the corresponding instruction pointer in the ICE, so that no re-compilation and no re-loading of the program is required during debugging.

Radiation effects in space environment

In the space, ionizing radiations have an adverse effects on electrical systems. Usually, a Single Event Upset (SEU) occurs when a single particle (i.e. a heavy ion or proton) deposits enough charge at

sensitive node in a microcircuit to cause that circuit to change state. In general, these effects are temporary and appears as “soft failures” such as anomalous bit flips or spurious commands. In extreme cases, latch up can occur and result in the destructive failure of the part when no special countermeasures are taken.

SPECIES system has two possibilities to prevent SEU failure: the first one is to use a radiation hardness technology which can limit the effects because of the process foundry and the second is to use special architectural design to protect registers and cell memories.

A standard SEU protection based on Triple Modular Redundancy (TMR) is implemented for internal registers.

The memory cells use an Error Detection And Correction (EDAC) strategy which is based on Hamming Coding so a Single Error Correction (SEC) and a Double Error Detection (DED) is performed.

Figure 4 shows the SPECIES address space from 00000_H to $3FFFF_H$ in four fixed blocks of 64 kByte for data memory. The program memory has 64 kByte and it can lie in one of four blocks.

The system contains a Built In Self Test (BIST) circuit connected to the internal memories and to the external memory too. Test procedures are common used for RAMs because they give a complete information about memory status which can be used to diagnose faults or to recover data from errors.

The external memory test is performed o*ff line*, that means that no program is running because the system is in a special test mode and the memory contain can be destroyed. An appropriate choice to run the BIST is at system startup.

The BIST subsystem implements the “Nair Abraham Thatte test” that with a medium complexity (30 *n*, where “*n*” is the number of cells) which detects a large set of possible faults: Single Stuck-At Faults (SAF), Address Faults (AF), Coupling Faults (CF) both between different cells and between different bits of the same cell.

An additional step of the algorithm allows to detect all the coupling faults between bits of the same cell. It is not in the original formulation. The step consists in writing special patterns “01…01…01…01…”, “0011…0011…”, “00001111…” and than checks the cell. The same operation is repeated with the inverted patterns.

When a fault occurs in a cell memory, it is possible to have information about a single or a double fault. In the first case the test continues because a special redundancy encoding based on Hamming Code in the stored data can correct a single error in the pattern and detect multiple errors in fact in the second case the test stops immediately.

The test result is available in a special status register: for each block we can obtain the following values:

00: test passed, the memory block is ok;

01: there are cells with single fault;

10: there are cells with multiple faults;

11: test found cells with single error and then cells with multiple errors.

The software application can decide if it is safe to use the memory in case of single error and to abort in case of multiple faults.

The interesting idea is to run BIST test before loading the code into the memory. The ICE block puts SPECIES in test mode to run the Nair Abraham Thatte test, then according to the test results the software can be relocated in one of the good memory banks. In this way the user, knowing the situation of the memory can decide which is the best part of memory were load the code, and eventually decide not to use certain sub-blocks where tests failed. Of course, a test can be repeated, but the system will be placed in test mode

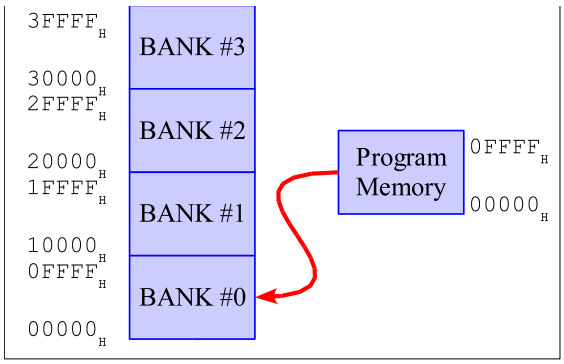

Figure 4 - The Address Space with Memory Map and the movable block for Program Memory

via the ICE block.

Testing and evaluation performances

The SPECIES system on chip was initially tested on a FPGA assembled in a prototype board with two 128kb x 8 flash memories and 256kb x 16 asynchronous RAM. An external cards with a CAN transceiver connected the system to a CAN network.

The communication link via the ICE module allowed to successfully download the software to SPECIES, then a Graphical User Interface (GUI) designed for this purpose, watched for the register contents and SFR access.

Some breakpoint were inserted in the code and the execution was successfully resumed. Moreover a data dump routine and text code disassembly showed the RAM memory contents. Test programs, written both in C and assembler, to test the 8051 instruction set, ALU arithmetic and logic operations and program control flow run fine [5]. The ACAN module and other peripherals could communicate with CPU and transmitting and sending messages into the CAN network.

Conclusions

The SPECIES system on chip integrates a CPU 8051-like with a CAN controller: it was designed to operate in hostile environments where the effects of radiation might cause a system failure. The internal registers and memories are protected against SEU. The BIST scans the memory cells to detect and correct errors; in case of unrecoverable error on program memory, it can swap the memory banks and relocate the code automatically.

The ICE allows to access the internal registers, memories and program execution flow for testing or debugging purpose.

References

[1] Department of Defense “Test Methods and Procedures for Microelectronics – MIL-STD-883.

[2] “CASA2 Technical Specifications” Aurelia Microelettronica CASA2/01/ CASA2/ SPT/_/4

[3] F. Hartwich, A. Bassemir, “The Configuration of the CAN Bit Timing” 6th International CAN Conference, 2nd -4th November 1999, Turin (Italy)

[4] MC51 Microcontroller Family User’s Manual, Intel, February 1994

[5] F. G. Lima, S. Rezgui, E Cota, L. Carro, M. Lubaszewski, R. Velazco, R. Reis “Designing and Testing a Radiation Hardened 8051-like Micro-controller”

Egidio Pescari
Aurelia Microelettronica
Via Giuntini 13 – 56023 Navacchio (Italy)
Phone: +39 050 754 260
Fax: +39 050 754 261
E-mail: Egidio.Pescari@Aurelia-Micro.it
Website: www.aurelia-micro.it

Andrea S. Brogna
Università degli Studi di Pisa
Dipartim. di Ingegneria dell’Informazione
Via Diotisalvi, 2 – 56122 Pisa (Italy)
E-mail: Andrea.Brogna@Aurelia-Micro.it

Roberto Saletti
Università degli Studi di Pisa
Dipartim. di Ingegneria dell’Informazione
Via Diotisalvi, 2 – 56122 Pisa (Italy)