DEBUGGING AND DIAGNOSTIC OF SENSORS CONNECTED IN A NETWORK USING MICROCONTROLLER MSP430

Georgi Dotsov Geshev, Emil Nikolov Dimitrov

In this paper is examined monitor program for debugging, adjustment and diagnostic of embedded microprocessor system based on TEXAS Instruments MSP430. It is foreseen to work with it while the separate systems are connected to a local network. After a review of the existing monitor programs and analysis of the available resources in the microcontroller family, functions are selected to be implemented into this service program. Some specifics and constraints are mentioned.

Keywords: Embedded system, microcontroller, monitor program

1. REVIEW OF EXISTING MONITOR PROGRAMS FOR MICROCONTROLLERS

In order to select appropriate command set to be implemented in the diagnostic program a review is made of the existing monitor program offered by popular 8- and 16-bit microcontrollers manufacturers. There are examined BUFFALO for HC11 [1], UltraMON51 for 8051 [2], QwickBUG for PIC18 [3]. All these programs have identical functions – load and print of memory arrays, special function registers review, breakpoint manipulation, trace, starting the user program. For the reviewed program connection between personal computer and the embedded system is peer-to-peer. The chosen microcontroller MSP430 is loaded and debugged using JTAG interface.

During creation of sensor and actuator systems composed of separate modules connected in a network, the tuning and initial set-up are more complicated. All the reviewed modules can not be used remotely in case the systems are connected in a local network. Additional functionality for debugging is added with the opportunity to implement all these functions remotely using certain network protocol for data exchange.

Due to the specifics of the different network technology the microcontroller's manufacturers do not offer ready to use network program modules. It is necessary to be defined and implemented such products for concrete microcontroller and support for certain network technology. Example for similar diagnostic program for HC11 is designed in the department of "Electronic systems" Technical University – Sofia [4].

2. MSP430 RESOURCES ANALYSIS

During analysis of the possibilities to implement certain functions, the architecture of the microcontroller, different addressing modes and the command set supported by the controller have to be taken into account. Typical for this family is: RISC architecture, von Neumann memory organization, 16 bit processor core, support of different addressing modes, few low power consumption modes.

In the particular family members is foreseen wide variety of supported periphery: 16 bit hardware multiplier, up to 12+2 bit SAR ADC, up to 16 bit σ-δ ADC, DAC using DMA controller and sophisticated timer system, up to 2 USART modules, I2C module [5]. All these advantages make MSP430 suitable for building up in a sensor and actuator systems.

3. NETWORK TECHNOLOGY

For data exchange between all modules in such sensor and actuator network have to be taken into account some specifics. The environment surrounding all these modules is noisy. In order not to loose data during exchange the physical layer must be noise resistant. The chosen network technology must support certain number of correspondents, connected in this network, big enough for the desired application. In the examined system is chosen differential half-duplex RS485 standard. It ensures noise protection and up to 32 correspondents.

Additional error protection could be achieved with the used transport protocol. The chosen one supports addressing, checksum error checking, alternating coloring of the messages in case the message is bigger than one package.

4. ALGORITHM OF THE MONITOR PROGRAM

Initialization – due to undefined main cycle time the WatchDog timer has to be disabled; serial communication port register is set according wanted baud rate; interrupt disabled; the used operational memory filled in with certain constants.

Sending of initial message to the main station.

Sending of a prompt.

Waiting for a capsulated message addressed to the particular module. In case there is no conformity of the message length or calculated checksum the message is ignored otherwise it is interpreted as a valid command.

The received message is sent to a command interpreter. It compares the command with the content of the written command table. In case of concurrence the control is given to particular subroutine for further data manipulation otherwise the control is returned back to the main program with an error message.

The subroutines manipulating the commands check if the input data are correct and performing the needed action.

There is only one requirement to the command interpreter of the user program. It is to support a command for service mode entering and executing the examined algorithm.

5. MONITOR PROGRAM IMPLEMENTATION

The structure of the program is given on fig. 1.

The service program has five relatively independent parts:

- Input-output communication buffer – it is a part of the operational memory big enough to store one capsulated message. Direct access to these data have only the input and the output systems. The common communication buffer for input

and output adds the constraint the next command not to be executed before the current one is still running.

- Input system – it follows the packages flowing through the network. If the current message is addressed to this module all checks of the message length and parity are performed. The input system gives to the other subroutines interface to access the input data.
- Output system – it capsulates the outgoing data according to a chosen protocol for data exchange. It provides also an interface to the other routines to write data in the input-output buffer.
- Main program cycle – the control is given to it when received message is correct capsulated and addressed to this module. Command interpreter decides if the received command exists in the table with supported commands. In case of correspondence appropriate subroutine is called for further data manipulating. Unrecognized command error message is sent otherwise.
- Subroutines – all parts of the program, which handle the received commands. Some of them are: memory initialization subroutine; subroutine for operational and program memory reading; subroutine for loading an array of data into the memory; breakpoint manipulation subroutines; system handling FLASH operations.

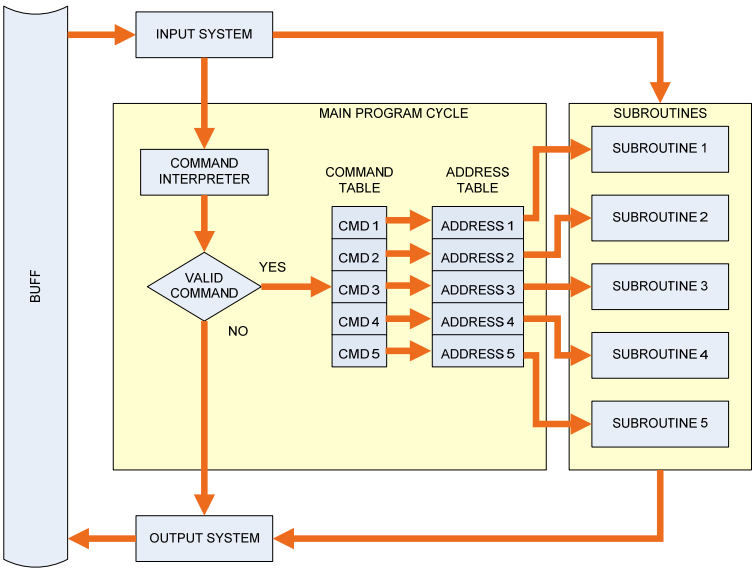

fig. 1 Service program structure

6. IMPLEMENTED DIAGNOSTIC FUNCTIONS

Certain message is sent to the system to enter diagnostic mode. The user has an opportunity to select a module using address.

Printing and initialization of memory regions. The user can read the existing program or constant table written in the controller memory.

Printing and set the contents of the microcontroller registers. The program counter, stack pointer, status register and 12 general purpose register contents can be modified. Registers R0 to R3 are special function registers and the user should be careful in case he wants to change their contents. Register R3 can not be modified.

Comparison and moving arrays of data. To the user is given a function, which can compare if the contents of two memory regions is equal. The user can move certain data from one place to another.

Write data array into the memory. This can be user program or the user can change certain parameters foreseen in the program. To develop this function certain constraints have to be taken into account – necessary operation time to write into FLASH memory, erasing of the entire page if it has to be reprogrammed.

All FLASH manipulating routines have to be considered to be compatible with FLASH specifics and constraints: FLASH page size is 512 Bytes for the segments in the main memory and 128 Bytes in the information memory; it can be erased only by pages; the logical state of erased bit is '1'; writing is comparatively slow process [6].

In the program is foreseen special algorithm for writing data into FLASH memory. Before writing each bit of the new data is compared with the existing one in the FLASH memory space. In case it is necessary to write a logical '1' into a place with logical '0' the entire page must be reprogrammed. A copy of this page is preserved in RAM. The new data is copied into RAM as well. After the new sector is formed in RAM memory the FLASH page can be erased and new data is copied word by word. After all these procedures acknowledgement has to be sent to the main station.

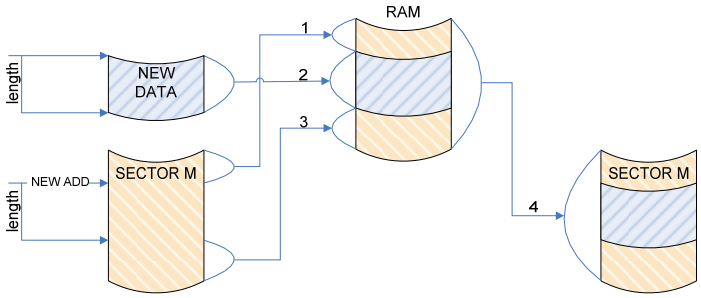

fig 2. FLASH page programming sequence

Those checks save time to the user because reprogramming of the whole page takes more time.

There is no access to the address and the command bus. This makes the implementation of certain functions very hard. Example is the trace function. It requires instruction disassembling, which takes a lot of resources, status register bits checking and next address calculation using different address modes (7 types of addressing are available). A balance has to be made between the implemented functions number, the program size and performance of the program. Sometimes we could replace tracing with few breakpoints.

Using similar approach with many breakpoint tracing could be implemented in the terminal program on the PC. There are no constraints about the size of the program. Even in this case should be taken into account that one trace execution is sequence of commands to be executed: instruction to be read, Status Register as well, calculating of the next address which could include some additional reading of registers or a field in the memory, to be written software interrupt on the new address.

This approach has some disadvantages:

- It is slow - it takes a lot of time for data exchange between the terminal program and the monitor program and reprogramming of FLASH memory takes additional time as well
- Each trace execution would rewrite two times one or two FLASH pages which could damage a FLASH cell after certain time of debugging using this approach

Breakpoint manipulation. This function is maybe the most valuable during debugging and tuning of a system. The user has an option to stop the execution of the program at a certain point and to print and/or change arrays of the program. The von Neumann architecture where the addresses and the data are in a single address space and the variable command length (1,2 or 3 words) oblige the user to choose address with command in it. Otherwise the chosen breakpoint can be interpreted as a data. The count of the manipulated breakpoints is dependent of the operational memory size. In the developed program manipulation with three breakpoints is foreseen. The user has possibilities to add a breakpoint, to remove a breakpoint, to remove all breakpoints and to print the contents of the breakpoint table.

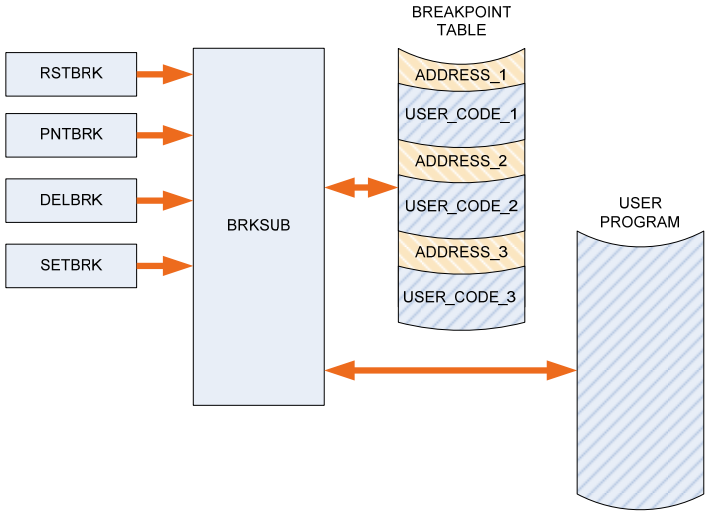

fig 3. Breakpoints subsystem structure

The breakpoint subsystem is built mainly of two components – breakpoint table and subroutines, manipulating defined breakpoints. The breakpoint table is reserved area in the operating memory where are written the breakpoints addresses and the temporary removed user code.

The subroutines are:

- Insertion in the user code – writing of specific information in the user program, which redirects the control to the breakpoint manipulation routine.
- Erasing a breakpoint – writing of already erased user code back into the user program.
- Removing a breakpoint – clearing the breakpoint from the table.

When the user program goes to a certain breakpoint the control is given to the breakpoint handling subroutine. Any type of software interrupt command is missing

in the command set of this microcontroller. The advantage of such interrupt routine is that the program counter, status register and some other special function registers could be saved in the stack automatically. All registers R0 to R15 must be saved without any data loss. Special attention has to be paid to registers R0 to R3 which are PC, SP, SR. In this case command "CALL(IMM)" is used. It saves PC and SP and does not affect any status bit. All the others general purpose registers are preserved as well.

When the user wants to continue execution of the program all registers except the program counter are copied back from the backup memory. The program counter is added on the top of the stack. After that "RET" (return from subroutine) command is used.

7. TERMINAL PROGRAM

To control this monitor program a personal computer program is needed. The PC program is a terminal for data exchange with the module trough the local network. When the terminal is connected to the sensor network the user can specify the address of the desired device. This software gives to the user interface to enter certain commands. There are no constraints concerning the program size, used memory and the performance of the terminal program in the comparison with the monitor program. Additional data manipulation is implemented. The syntax of the commands is checked using monitor program specifics and constraints. Some initial reformatting of the commands is performed. If the size of the array which the user wants to print or to write exceeds the size of the protocol package it is fragmented into smaller pieces. All these manipulations are hidden for the end user of the terminal program.

Physical layer of communication between microcontroller system and PC is industrial network using RS485. This protocol ensures noise protection of the transferred data. In the used transport level protocol some additional error checking mechanisms are foreseen.

This diagnostic program is implemented and tested on MSP430F149. The program size is less than 2 kB. For proper work of this program the used controllers must be with at least 1 kB RAM memory. This constraint is forced by the flash routines where entire FLASH page (512 Bytes) have to be written into the RAM memory.

The performance of the system depends of the communication speed.

8. REFERENCES

[1] BUFFALO MONITOR for HC11 Development Boards. / buffalo.pdf /

[2] Sims B. QWICKBUG MONITOR PROGRAM / reviseda4.pdf /

[3] UltraMON51 / ultramon.doc /

[4] Dimitrov E., G. Mihov, I. Tashev, M. Mitev. Local Array Network for Industrial Controller. The International Scientific Conference ENERGY AND INFORMATION SYSTEMS AND TECHNOLOGIES, vol. III. pp. 608-613. Bitola, Macedonia, 2001.

[5] "MSP430x1xx Family" Texas Instruments User's Guide. / slau049d.pdf /

[6] MSP430 Flash Self-Programming Technique / slaa103.pdf /