BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 64 (68), Numărul 1, 2018
Secția
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

MANAGING DYNAMIC RANDOM ACCESS MEMORY BANKS USING AN ATMEL MICROCONTROLLER

BY

PETRUȚ DUMA*

Technical University “Gheorghe Asachi” of Iaşi,
Faculty of Electronics, Telecommunications and Information Technology

Received: February 27, 2018
Accepted for publication: March 30, 2018

Abstract. The paper describes the hardware structure of the data read/write interface for high capacity dynamic random access memory banks, as well as the way to accomplish the refresh while the microcontroller reads the codes of the performed instructions. An ATMEL microcontroller-based development system controls the dynamic memory, various sensors and transducers as well as a real-time clock. The high volume of data acquired over short periods of time is stored in the memory and then loaded into a personal computer. The command program, written in machine code, implements a set of commands that test the memory, display the acquired data, substitutes data, loads, erases, moves, sends/receives data blocks etc.

Key words: dynamic random access memory; refresh process; address multiplexing; ATMEL microcontroller.

1. Introduction

The Dynamic Random Access Memory (DRAM) chips use electrical charge storage and transfer in order to store binary information. They use CMOS technology, are based on charging/discharging of a capacitor and work in pulses. Each memory cell consists of integrated switching transistor and a low value capacitor.

*Corresponding author: *e-mail*: pduma@etti.tuiasi.ro

A memory circuit includes the DRAM memory cells array, a row and a column decoder, input/output amplifiers used for reading/writing data, data multiplexer/de-multiplexer circuits, registers, command and control logic and a refresh controller. The latter can be absent from some circuits.

The memory cells are organized in a square or rectangular area, and each location stores a 1-bit, 4-bit, 8-bit or even more bits word.

DRAM memory banks use an address multiplexing technique in order to reduce the number of exterior connections for the integrated circuit capsule. The command signals load successively the row and column registers and perform commands in accordance with the addresses stored in the row and column decoders.

Since the information is basically stored in capacitors, that tend to discharge in time, reading and writing the memory cells repeatedly are required. This represents the refresh process, necessary for the DRAM memory banks, that takes from a few milliseconds up to tens or even hundreds of milliseconds. It is performed for all the cells in a row simultaneously and it requires a systematic approach for the entire memory circuit.

The basic structure of a system for acquiring data from various sensors and storing the measured data in a DRAM memory bank, using a development system equipped with an ATMEL family microcontroller, is shown in Fig. 1. The notes have the following meaning: S_1, S_2, S_3, … S_K – sensors (digital/ analogue); SI – sensor interface; RTC – real-time clock; DS_ATMEL_µC – development system using an ATMEL microcontroller; DRAM_I – DRAM interface; B_i_DRAM – bank DRAM (i = 0, 1, 2,…); SI_RS232 – serial interface RS232; PC –personal computer.

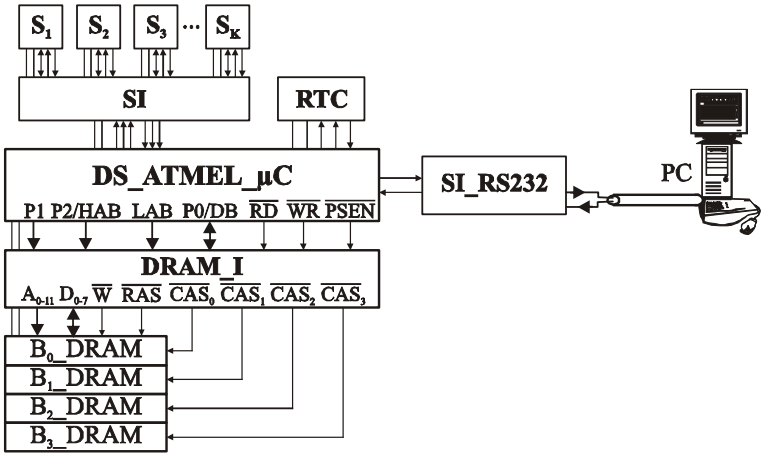

Fig. 1.

The DRAM memory is loaded with the values of the quantities measured from sensors at user-programmed time intervals, along with the real

time clock value. The development system communicates serially with a personal computer, in order to load the acquired data, but also for testing the memory, displaying data and other commands.

The present paper covers the interfacing the development system with several high capacity asynchronous DRAM memory banks. The refresh operation is made by the implemented interface by enabling only the row address strobe signal.

2. Interfacing DRAM Memory Banks

The DRAM memory consists at this first stage of a bank containing eight TC514100 circuits, having a 4Mbits capacity each. Subsequently, more memory banks can be added, each bank being selected by the outputs of a decoder. Alternately, memory circuits with different capacities may be used.

The microcontroller-based development system addresses the external memory (EM) within the address space 0000-7FFFH (BA_{15}=0) and consists of a non-volatile memory (FLASH) and a volatile memory (SRAM) that allow the execution of program instructions, data reading and data writing respectively.

The address space 8000H-0FFFFH is used only for data read/write into the external DRAM memory (BA_{15}=1). The DRAM memory has a 4 MB capacity, organized for selection in 128 pages of 32KB each.

The microcontroller's command signals for instruction code read ($\overline{PSEN}$), data read ($\overline{RD}$), data write ($\overline{WR}$) and the address line BA_{15} validate the external memory circuits (EM and DRAM) and perform on these memory areas the operations shown in Table 1.

Table 1

$\overline{PSEN}$	$\overline{RD}$	$\overline{WR}$	BA_{15}	Feature
1	1	1	0	–
1	1	1	1	–
⎽⎍⎽	1	1	0	EM instruction code read
⎽⎍⎽	1	1	1	–
1	⎽⎍⎽	1	0	EM data read
1	⎽⎍⎽	1	1	DRAM data read
1	1	⎽⎍⎽	0	EM data write
1	1	⎽⎍⎽	1	DRAM data write

The Tc514100 circuits, used to implement a single 4MB memory bank, are connected as in Fig. 2. The corresponding address lines A_0, A_1, … A_{10} of the memory circuits are connected together to the respective outputs of the address multiplexers. the data input line (D_{in}) is connected to the data output line (D_{out}) of each memory circuit and represents the data line of the respective memory circuit (D_j, j = 0, 1, … 7).

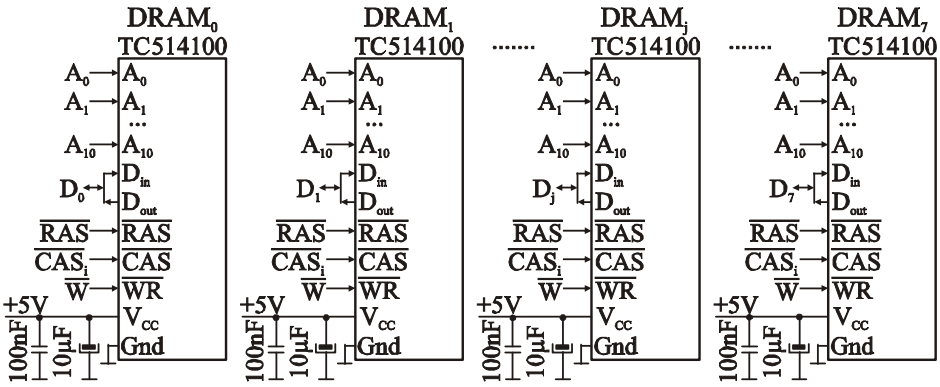

Fig. 2.

The $\overline{\text{RAS}}$ command lines of the memory circuits from all the banks are connected together and represent the row address strobe signal. The $\overline{\text{CAS}}$ command lines of every 8 memory circuits bank are connected together and represent the column address strobe signal ($\overline{\text{CAS}_i}$). The write lines $\overline{\text{WR}}$ from all the memory circuits are connected together to one single line (noted $\overline{\text{W}}$) and represents the signal determining a data read operation if $\overline{\text{W}}$ =1 or a data write operation if $\overline{\text{W}}$ = 0. Each memory circuit has also connected as close as possible to the power terminals (V_{CC} and Gnd), a group of capacitors, consisting of a 100nf ceramic capacitor and a 10 µf tantalum electrolytic capacitor, used for filtering and de-coupling.

Multiplexing the DRAM addresses is made with six 74S153 Circuits (Mux_k, k = 0, 1,…5) as in Fig.3.

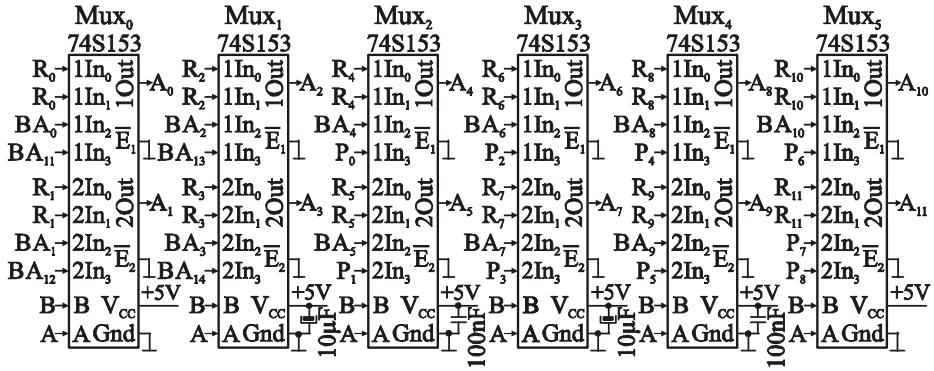

Fig. 3.

Each integrated circuit has two multiplexers with four inputs and an output. The enabling inputs of the multiplexers are constantly active ($\overline{E}_1 = 0$ and $\overline{E}_2 = 0$). The corresponding selection inputs of these circuits, B and A respectively, are connected together in order to provide at the outputs of the multiplexers the inputs of the same rank.

At the multiplexers' inputs In_0 and In_1 respectively, the refresh addresses R_0, R_1, … R_{11} generated by the refresh counter are connected. At inputs In_2 of the multiplexers, the row address is connected. It consists of the 11 least significant address lines of the 32KB memory page from the DRAM memory, i.e. BA_0, BA_1, … BA_{10} from the microcontroller's address bus. Inputs In_3 of the multiplexers are connected to the column address, that consists of the 4 most significant address lines of the 32KB memory page of the DRAM memory, *i.e.* BA_{11}, BA_{12}, BA_{13}, BA_{14} from the microcontroller's address bus and the memory page selection address lines P_0, P_1, … P_6, provided by a serial-parallel register loaded through software.

The multiplexers' outputs (Out) transmit on the DRAM memory address lines A_0, A_1, … A_{10}, either the refresh address or the row address or the column address, depending on the selection inputs B and A, as in Table 2.

Table 2

B	A	Multiplexers' outputs – DRAM address											Feature
0	0	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	Refresh address
0	1	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	Refresh address
1	0	BA_0	BA_1	BA_2	BA_3	BA_4	BA_5	BA_6	BA_7	BA_8	BA_9	BA_{10}	Row address
1	1	BA_{11}	BA_{12}	BA_{13}	BA_{14}	P_0	P_1	P_2	P_3	P_4	P_5	P_6	Column address

The refresh address R_0, R_1, … R_{11} is provided by a 12 bit counter implemented using three four-bit synchronous 76161 counters, as in Fig.4.

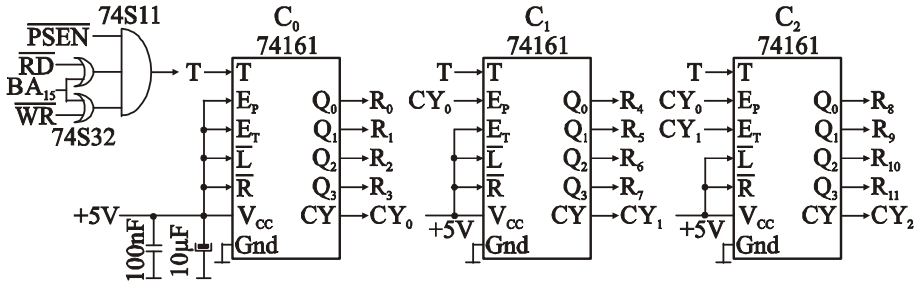

Fig. 4.

The carry output CY_0 of counter C_0 commands the enable input E_P of counters C_1 and C_2, while the carry output CY_1 of counter C_1 commands the

enable input E_T of counter C_2. The other unused enable inputs of the counters, the parallel load ($\overline{L}$) and the initialize ($\overline{R}$) are inactive (logical 1).

The clock inputs T of the three counters are connected together and controlled by the signal:

$$T = \overline{\overline{PSEN} \cdot (\overline{RD} + BA_{15}) \cdot (\overline{WR} + BA_{15})}$$

The clock signal T becomes logical 0 during the read cycles of the instruction codes from the external program memory and also during read and write cycles of data in the external data memory (EM). After the execution of the respective cycle, T becomes logical 1, while on the rising edge of the clock signal, the refresh counter is incremented by one unit (Table 3). During DRAM read and write cycles, the clock signal remains logical 1 and the refresh counter is inactive (Table 3).

Table 3

$\overline{PSEN}$	$\overline{RD}$	$\overline{WR}$	BA_{15}	T	Feature
1	1	1	0	1	–
1	1	1	1	1	–
⎍	1	1	0	⎍	Refresh counter increment
⎍	1	1	1	⎍	Refresh counter increment
1	⎍	1	0	⎍	Refresh counter increment
1	⎍	1	1	1	Refresh counter inactive
1	1	⎍	0	⎍	Refresh counter increment
1	1	⎍	1	1	Refresh counter inactive

The DRAM command signals $\overline{RAS}$ and $\overline{CAS}$ and the multiplexer selection signals B and A are generated by the logical structure shown in Fig. 5.

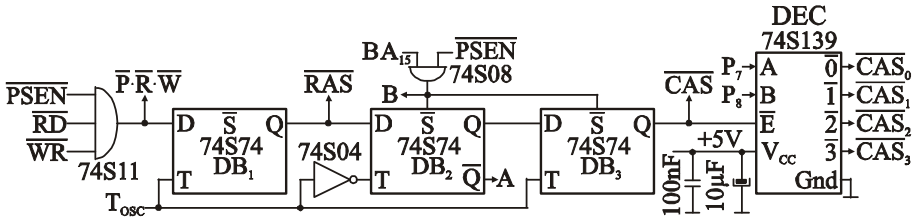

Fig. 5.

A logical AND gate having signals $\overline{PSEN}$, $\overline{RD}$ and $\overline{WR}$ as inputs produces the access signal for any external memory location ($\overline{P} \cdot \overline{R} \cdot \overline{W}$). This signal, delayed with $T_{OSC}/2$ by a D-type bistable circuit (DB_1) delivers the command signal $\overline{RAS}$.

The B selection signal is obtained by a logical AND gate using $\overline{PSEN}$ signal and address line BA_{15} as inputs. B is logical 0 either when the instruction

code is read, or when a data is read or written in EM. B turns to logical 1 when data are read or written in DRAM.

The A selection signal is obtained using a D-type bistable circuit (DB_2) from the $\overline{RAS}$ signal delayed by $T_{OSC}/2$, inverted and conditioned by signal B.

If B is logical 0, the DB_2 bistable is set and A becomes logical 0; the multiplexers provide the refresh address at their outputs.

If B turns to logical 1, then A is logical 0, and the multiplexers provide the row address at their outputs. After activating signal $\overline{RAS}$, signal A becomes logical 1 with a $T_{OSC}/2$ and then the multiplexers provide the column address.

The command signal $\overline{CAS}$ is obtained from the selection signal A delayed by $T_{OSC}/2$ using a D-type bistable circuit (DB_3), conditioned also by signal B.

If B is logical 0, A is also logical 0, bistable circuits DB_2 and DB_3 are set, therefore signal $\overline{CAS}$ being inactive (logical 1). In this case, only $\overline{RAS}$ signal is active, performing the DRAM refresh operation (Table 4).

If B becomes logical 1, then A being logical 0, and the DRAM address lines contain the row address that will be strobed as $\overline{RAS}$ is activated. After a $T_{OSC}/2$ delay, signal A turns into logical 1, while the DRAM address lines contain this time the column address to be strobed as $\overline{CAS}$ becomes active. In this case, a data read or write operation is performed in DRAM (Table 4).

Table 4

$\overline{PSEN}$	$\overline{RD}$	$\overline{WR}$	BA_{15}	$\overline{P}\,\overline{R}\,\overline{W}$	$\overline{RAS}$	B	A	$\overline{CAS}$	Feature
1	1	1	0	1	1	0	0	1	–
1	1	1	1	1	1	1	0	1	–
⎍	1	1	0	⎍	⎍	0	0	1	DRAM refresh
⎍	1	1	1	⎍	⎍	0	0	1	DRAM refresh
1	⎍	1	0	⎍	⎍	0	0	1	DRAM refresh
1	⎍	1	1	⎍	⎍	1	**0/1**	⎍	DRAM data read
1	1	⎍	0	⎍	⎍	0	0	1	DRAM refresh
1	1	⎍	1	⎍	⎍	1	**0/1**	⎍	DRAM data write

The outputs of decoder 74S139 (DEC) select each a DRAM memory bank using signal $\overline{CAS_i}$ (i =0, 1,…,3) if a multi-bank high capacity memory is used.

The DRAM data lines are passed through a 74S245 bidirectional buffer in order to be connected to the microcontroller's data bus, as in Fig.6. Thus, the DRAM data lines (D_0, D_1,…,D_7) are connected to inputs/outputs A^*_0, A^*_1,…,A^*_7 of the buffers, while the microcontroller's data lines (BD_0, BD_1, … BD_7) are connected to the outputs/inputs B^*_0, B^*_1,…,B^*_7 of the bi-directional buffers.

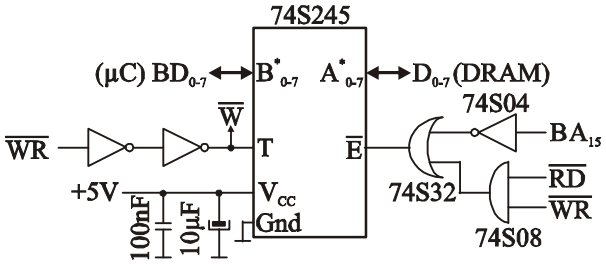

Fig. 6.

The bi-directional buffer is enabled by the signal $\overline{E}$ which is active when data is either read ($\overline{RD}$=0 and BA_{15}=1) or written ($\overline{WR}$ and BA_{15}=1) in DRAM.

$$\overline{E} = \overline{RD} \cdot \overline{WR} + \overline{BA}_{15}$$

The data transfer direction through the bi-directional buffer is determined by the command signal for data write into the memory ($\overline{WR}$).

The address of the DRAM memory pages is provided by a 74164 serial-parallel register, connected as in Fig. 7.

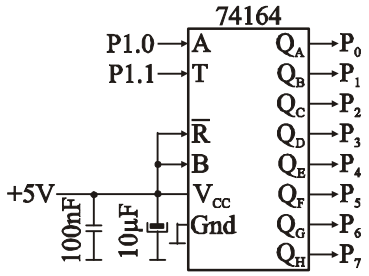

Fig. 7.

The eight-bit register provides at the address of the selected DRAM memory page (P_0, P_1, … P_7) at outputs Q_A, Q_B, … Q_H. The register's A data input and clock input T are connected to lines P1.0 and P1.1 respectively in order to allow the software command from the microcontroller when the memory page address update is required. The initialization input ($\overline{R}$) and the data input B of the register are inactive (logical 1).

For DRAM circuits with a capacity equal to or higher than 16MB, a serial-parallel 16-bit register is required, implemented with two 74164 circuits.

The general time diagram of the command signals used for executing instructions from the external program memory of the microcontroller-based development system is presented in Fig. 8.

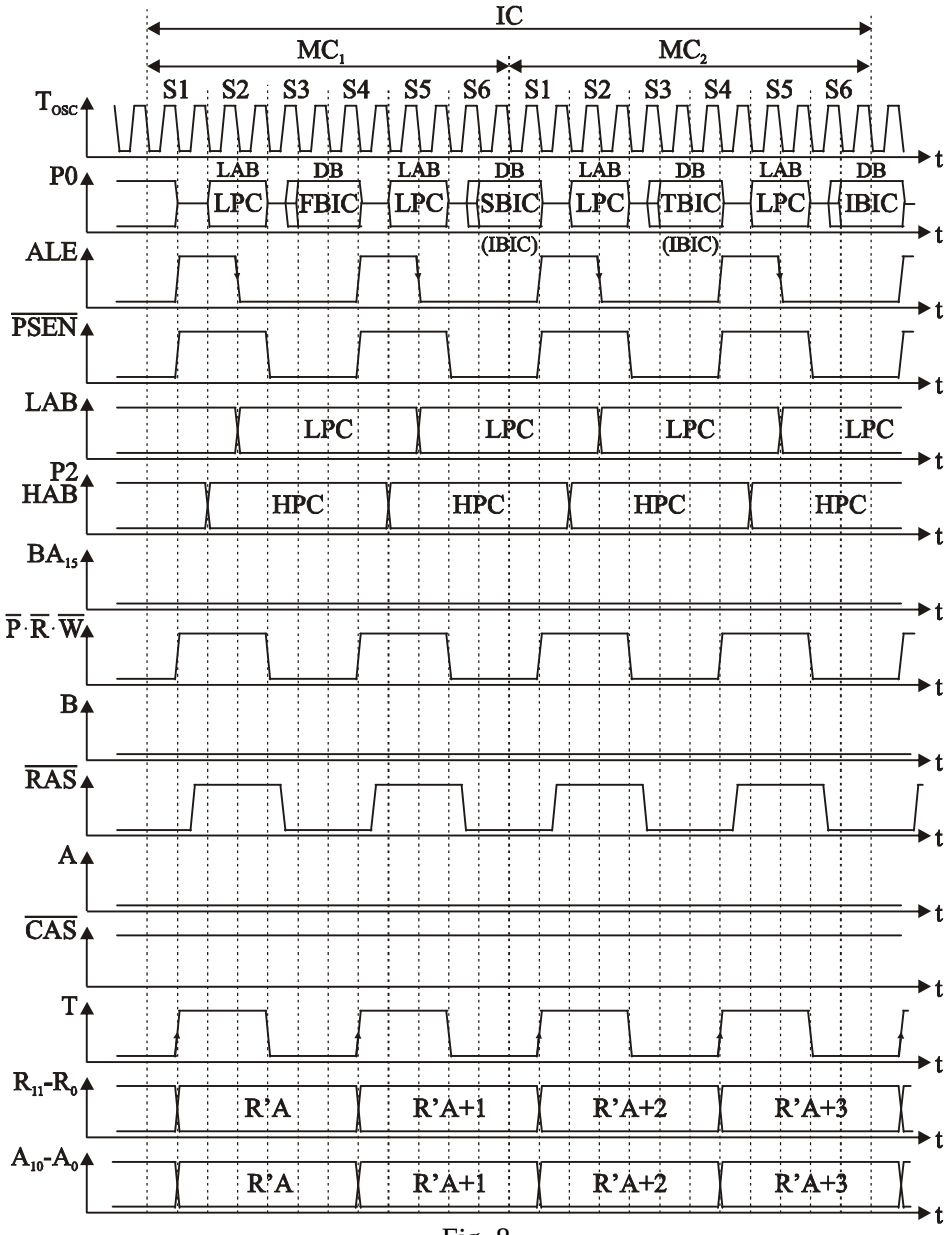

Fig. 8.

The notes have the following meaning: IC – instruction cycle; MC1, MC2 – machine cycles; LAB – lower part of the address bus; HAB – higher

part of the address bus; LPC – lower part of the program counter; HPC – higher part of the program counter; FBIC – the first byte of the instruction code; SBIC – the second byte of the instruction code; TBIC – the third byte of the instruction code; DB – data bus; R'A – refresh address; S1, S2, … S6 – the machine cycle stages; IBIC – ignored byte from the instruction code. This diagram does not cover the instructions performing data read or write from/into the external data memory.

The external program memory is read in each half machine cycle. Port P0 provides the low part of the address from the program counter during stages S2 and S5 respectively, from each machine cycle. On the falling edge of the ALE signal, this address is memorized in an external latch, in order to allow for the bus de-multiplexing. During the rest of the cycle, port P0 is a data bus for reading the instruction bytes. Port P2 provides during stages S2, S3 and S4, and during stages S5, S6 and S1from every machine cycle, the high part of the address in the program counter.

The command signal $\overline{PSEN}$ is activated from the beginning of stage S3 until the middle of stage S4 and from the beginning of the stage S6 until the middle if the next stage S1. This signal is used for enabling the external program memory in order to read the current byte of the instruction currently being executed.

The first read byte is the instruction code, while the following represent a data, an address, an offset etc. The microcontroller instructions are one, two or three bytes long and are executed in one, two or four machine cycles. Some instructions require a longer time for execution, in which case the following instruction code is read and ignored, while the program counter is not incremented.

During each half machine cycle, while the current instruction code is being read, the DRAM is also refreshed. The address line BA_{15} is logical 0 and the selection signals B and A of the multiplexers are both also logical 0. The address lines of the DRAM will contain the refresh address. After activating signal $\overline{PSEN}$, $\overline{RAS}$ is also active after a $T_{OSC}/2$ delay, while signal $\overline{CAS}$ stays inactive. Thus, as the microcontroller reads and executes instruction, a line by line DRAM refresh is performed at the same time. After $\overline{PSEN}$turns inactive, on the falling edge of the clock signal T the refresh counter is incremented by one unit.

The time diagram of the command signals used for reading data from DRAM is shown in Fig. 9; the notes have the following meaning: DPL – lower part of the DPTR register; DPH – higher part of the DPTR register; R_j – general register R_0 or R_1; RA – row address; CA – column address.

These instructions have a one-byte length and are executed in two machine cycles. During the first three stages of the first machine cycle of these instructions, the code of the instruction which is to be executed is read, while during the last three stages of the second machine cycle, the code of the following instruction is read and discarded. During these half cycles the DRAM refresh is also performed, as described in the previous paragraphs. During the

second part of the first machine cycle the address of the memory location is updated and in the first part of the second machine cycle, the DRAM data read operation is actually performed.

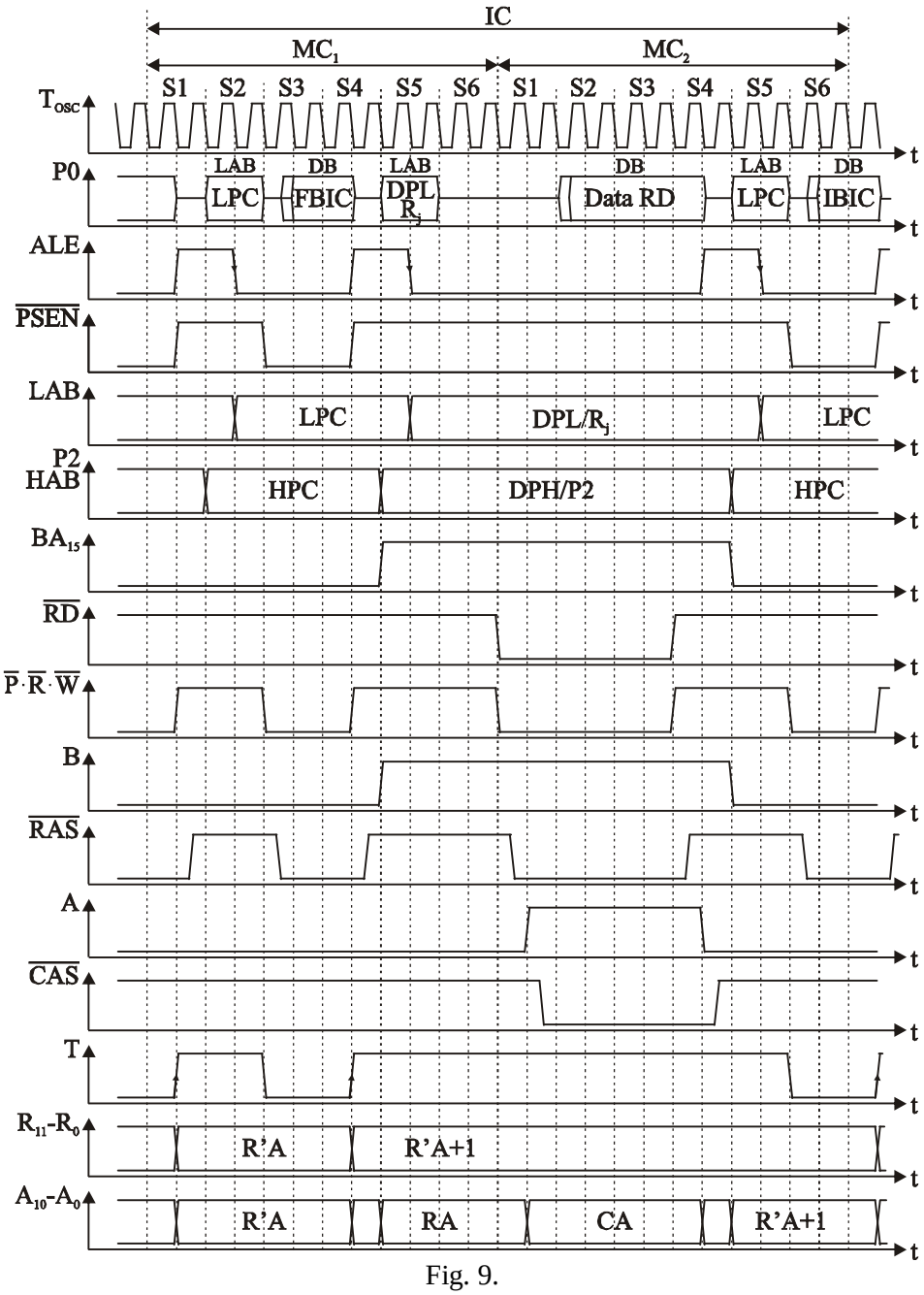

Fig. 9.

During stage S5 of the first machine cycle, the low part of the address corresponding to the data memory location to be read is delivered on port P0. O the falling edge of signal ALE, this address is stored in an external latch used for bus de-multiplexing. Port P2 provides, from the beginning of stage S5 of the first machine cycle until the end of stage S4 of the second machine cycle, the high part of the address corresponding to the data memory location to be read.

During stages S1, S2 and S3 of the second machine cycle, the $\overline{RD}$ signal is activated, and port P0 becomes data bus.

The address line BA15 is logical 1 from the second part of the first machine cycle, while the selection signals B and A of the multiplexers are logical 1 and logical 0 respectively. The DRAM address lines hold the row address of the memory location to be read. At the beginning of the second machine cycle, the read signal $\overline{RD}$ is activated, which will determine the activation of the $\overline{RAS}$ signal with a $T_{OSC}/2$ delay and the storing of the row address into the DRAM. The selection signal A of the multiplexers becomes logical 0 after a further $T_{OSC}/2$ delay. In this case, the DRAM address lines hold the column address of the memory location to be read. In the end, after a new $T_{OSC}/2$ delay, the $\overline{CAS}$ signal is activated and the column address is stored the DRAM. After the memory access time has passed, the data bus lines (port P0) contain the data read from DRAM, which will be loaded into the microcontroller.

The time diagram with the command signals for writing data into the DRAM is shown in Fig. 10. Similar to the data read instructions, the data write instructions are one byte long and are executed through two machine cycles.

The evolution of the signals during a DRAM data write is also similar to that described for the DRAM data read. The first machine cycle and the last three stages of the second machine cycle are identical for the two types of instructions. The differences appear at the end of the first machine cycle and during the first three stages of the second machine cycle and they are described below.

Port P0 provides the data that is to be written into the DRAM and remains unchanged from the beginning of stage S6 of the first machine cycle until the end of stage S3 of the second machine cycle. During stages S1, S2 and S3 of the second machine cycle, the write signal $\overline{WR}$ is activated. This signal triggers the DRAM write operation. The address line BA_{15} is logical 1, and the multiplexers' selection signals B and A, and the command signals $\overline{RAS}$ and $\overline{CAS}$ respectively, have an evolution similar to the case of reading data from DRAM, with the adequate modifications.

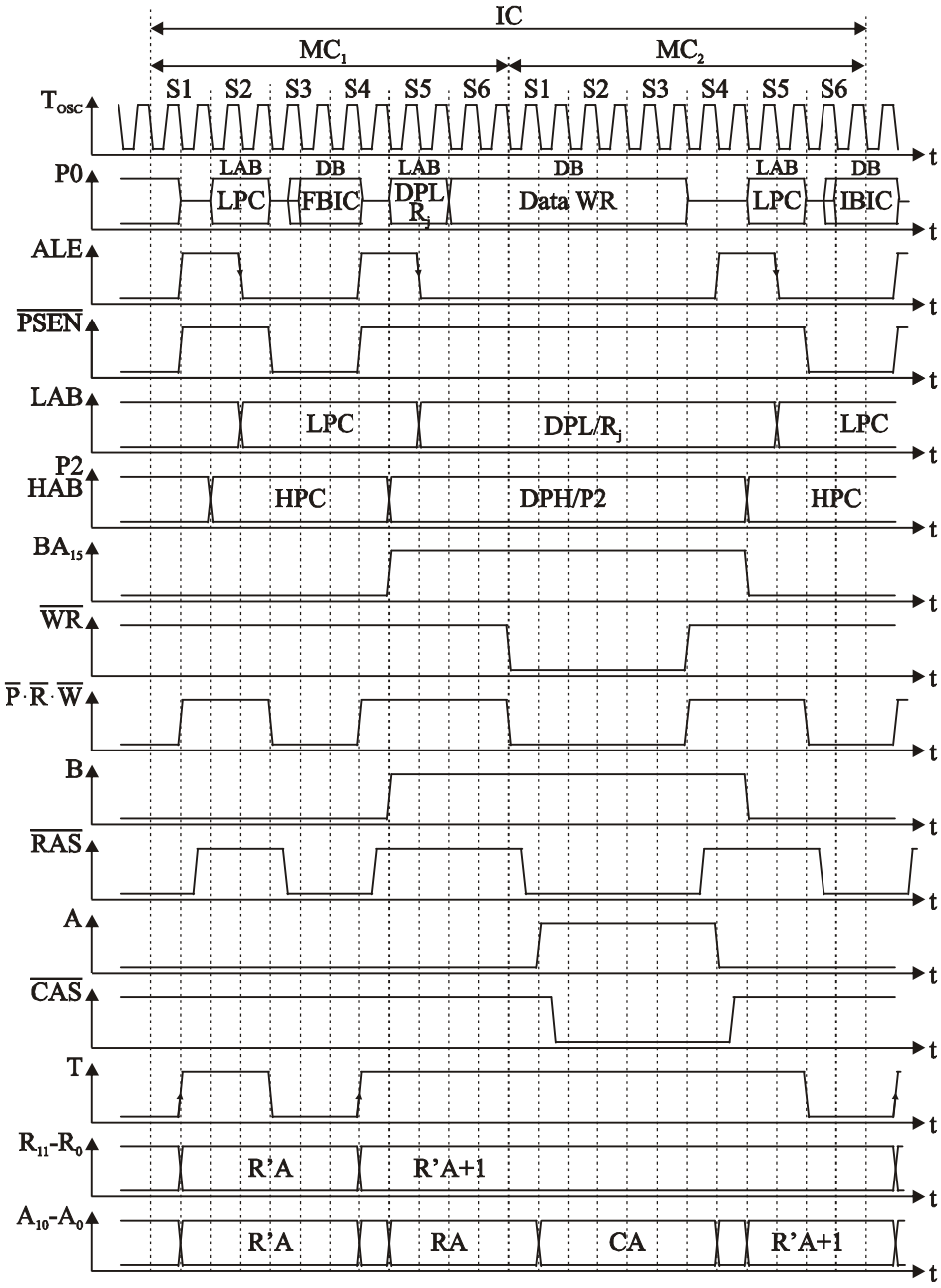
IC
MC1
MC2
S1 S2 S3 S4 S5 S6 S1 S2 S3 S4 S5 S6
Tosc
P0
LAB DB LAB DB LAB DB
LPC FBIC DPL Rj Data WR LPC IBIC
ALE
PSEN
LAB
LPC DPL/Rj LPC
P2
HAB
HPC DPH/P2 HPC
BA15
WR
P R W
B
RAS
A
CAS
T
R11-R0
R'A R'A+1
A10-A0
R'A RA CA R'A+1
t

Fig. 10.

4. Conclusions

The hardware structure described, used for interfacing asynchronous DRAM circuits, was built in practice. It consists of one or several memory banks, the multiplexing circuits for row, column and refresh addresses, the address counters, the logic circuitry for generating command signals, the data bus driver and the serial-parallel register of the page address.

The DRAM memory banks have a high capacity, a low energy consumption, a low cost per bit and access time in the range of 70 ns – 100 ns – 150 ns (somewhat high); the memory banks require a refresh circuit, fact that represents a disadvantage.

The interface is managed by a development system equipped with an ATMEL microcontroller, operating at an internal clock frequency from 12 MHz up until 24 MHz/40 MHz.

This memory is required for acquiring large data volumes in low time intervals from various sensors that are also commanded and controlled by the microcontroller.

The DRAM memory banks may be implemented with different circuit types with 1 MWord, 4 MWord or 16 MWord capacities, using either 1-bit, 4-bit or 8-bit words.

The software that was developed implements commands that allow memory testing, memory data display in various formats, data substitution, loading data in a personal computer as a text file database etc. The command program was written in machine code, uses only 3.8 KB of memory area, a small amount of memory compared to the possibilities it offers and the features implemented**.**

REFERENCES

Aghion C., Ursaru O., *Aplicații practice ale microcontrolerelor*, Edit. PIM, Iaşi, 2009.

Aghion C., Ursaru O., *Informatică aplicată. Introducere în microcontrolere*, Edit. PIM, Iaşi, 2015.

Balan R., *Microcontrolere. Structură şi aplicații*, Edit. Todescu, Cluj-Napoca, 2002.

Baruch Z., *Tehnologii de memorii DRAM*, NET Report, Vol.11, nr.118-120, 2002.

Duma P., *Microcontrolere în telecomunicații*, Edit. TEHNOPRESS, Iaşi, 2007.

Peatmann B.J., *Design with Microcontrollers*, McGraw Hill, New York, 1998.

Petreuş D., Muntean G., Juhos Z., Palaghița N., *Aplicații cu Microcontrolere din Familia 8051*, Ed. Mediamira, Cluj-Napoca, 2005.

Popa M., *Proiectarea Microsistemelor Digitale*, Edit. Orizonturi Universitare, Timişoara, 2003.

Oniga Șt., *Circuite digitale*, Edit. Risoprint, Cluj Napoca, 2002.

* * * *Family Microcontroller*, ATMEL, Data Book, 1998.

* * * *CMOS Memory*, ATMEL, Data Book, 2004.

* * * *AT89S8253 Microcontroller*, ATMEL, Data Sheet, 2005.

* * * *HM5116100F, HM5117800F, 16Mbit DRAM*, HITACHI, Data Sheet, 1997.

* * * *HM5164805F, HM5165805F, 64Mbit DRAM*, HITACHI, Data Sheet, 1999.

* * * *HM5112805F, HM5113805F, 128Mbit DRAM*, HITACHI, Data Sheet, 2000.
* * * *Multichannel RS232 Drivers/Receivers*, MAXIM, MAX232A Data Sheet, 2006.
* * * *MOS Memory Commercial and Military*, Texas Instruments, Data Book, 1995.
* * * *Digital Logic*, Texas Instruments, Data Book, 2007.
* * * *TMS44400, TMS46400 Dynamic Random Access Memories*, Texas Instruments, Data Sheet, 1996.
* * * *TMS416100, TMS416400 Dynamic Random Access Memories*, Texas Instruments, Data Sheet, 1995.
* * * *MOS Memory Products*, TOSHIBA, Data Book, 1998.
* * * *TC514100 DRAM*, TOSHIBA, Data Sheet, 1996.

GESTIONAREA BANCURILOR DE MEMORIE DINAMICĂ CU ACCES ALEATOR UTILIZÂND UN MICROCONTROLER ATMEL

(Rezumat)

Lucrarea prezintă structura hard a interfeței de citire/scriere a datelor în bancurile de memorie dinamică cu acces aleator, de capacitate mare, dar şi modul în care se realizează refresh-ul acestor memorii atunci când microcontrolerul citeşte codurile instrucțiunilor pe care le execută. Un sistem de dezvoltare cu microcontroler ATMEL gestionează memoria dinamică, diferiți senzori şi traductoare şi un ceas de timp real. Datele achiziționate în volum mare şi în intervale de timp scurte sunt salvate în memorie, apoi încărcate într-un calculator personal. Programul de comandă scris în limbaj maşină implementează o serie de comenzi care testează memoria, afişează datele achiziționate, substituie datele, încarcă, şterge, mută, transmite/recepționează blocuri de date şi altele.