

Int 15 emulator driver for Windows.

1. Int15 Hardware

Resources:

1. EEPROM - 2K size
000h-3FFh - reserved
400h-7FFh - free for user data
2. Temperature sensor
3. Watchdog hardware.
4. Extended digital I/O functions (MSEP945 board only)

Access to these resources under DOS can be provided by INT 15h function, see user manual, except Extended digital I/O functions.

Access under Windows XP, Windows Vista/7 can be provided over "Int15dl"-WDM driver.

For the moment this driver support all Kontron Compact Computers boards with PIIX4 and ICH4, ICH6, ICH7, ICH9, SCH, Stellarton chipsets and LX800.

2. Driver Installation (Windows-XP/Vista/7 32/64 bits)

Run "Int15Install41.exe", click "Next" button. Wait till driver installation process will be finished. Then press "Close" button.



Note: For a driver installation under Windows Vista/7 user must have a privileges of administrator.

3. Programming Int15dl interface under Windows:

Programming of the Int15dl Interface is very similar to DOS programming, based on DeviceIoControl function, which operate with predefined structure named "Registers".

Files:

Int15srv.h contains definitions for Registers structure.

Int15dlioctl.h contains definition for IO control code constants.

Test_Int15dl.cpp - example of subroutines, which provide access to hardware functions over Int15dl driver.

Functions (Test_Int15dl.cpp):

bool Int15(TRegisters *Regs) - the main function, which send user request to driver.

Returns **true**, if request finished successfully, otherwise **false**.

Regs - address of TRegisters structure, which contains specific data of request (defined in Int15srv.h).

For example, the following code will initiate temperature measuring:

```
TRegisters Regs;
Regs.ah = 0x78EC;
if(!Int15(&Regs)) //error in driver request
{
    printf("Error reading temperature\n");
    return;
}
//success - temperature value is in Regs.al
if(Regs.bl == 0)printf("\tTemperature = %d C\n",Regs.al);
//error - not valid value
else printf("\tError reading Temperature\n");
```

Note: Input and output arguments of Int15 function for different chipsets and BIOSes are different, please, read the user manual about registers definition. For example: To get temperature value on the board with PIIX4 chipset you have to use "Regs.ah = 0xEC;", but on the board with ICH4(6), ICH7 chipset and LX800, SCH, please use "Regs.ax = 0x78EC;".

bool Open_Int15dl(void) - the first function, which must be called to create a link between "Digital-Logic INT15 functions emulator" driver and user software. It returns **"true"**, if device was successfully opened, otherwise - **"false"**.

void Close_Int15dl(void) - the last function, which breaks a link between driver and user software.

int GetChipID(void) - additional service function, returns the type of chipset - for PIIX4 = 4, for ICH4(6) = 5, ICH7 = 9, LX800 = 7, SCH = 10, ICH9 = 11, Stellarton = 12.

Extended functions for SIO Winbond 83627 configuration (read Winbond 83627 datasheet):

BYTE SIO_In(BYTE Index) and
void SIO_Out(BYTE Index, BYTE Val) – access to SIO “Global” registers .

BYTE SIODev_In(BYTE Dev, BYTE Address) and
SIODev_Out(BYTE Dev, BYTE Address, BYTE Val)– access to SIO “Device” registers .

Extended functions for a second SIO Winbond 83627 registers
(only for MSEP945 board – Digital I/O):

BYTE SIO2_In(BYTE Index) and
void SIO2_Out(BYTE Index, BYTE Val) – access to SIO2 “Global” registers .

BYTE SIO2Dev_In(BYTE Dev, BYTE Address) and
SIO2Dev_Out(BYTE Dev, BYTE Address, BYTE Val)– access to SIO2 “Device” registers .

Before use any physical access to extended Digital I/O functions, the hardware has to be initialized as it written in the next example:

```
if(!Open_Int15dl())//if it's not successful, Int15 driver is not installed
{
    printf("ERROR opening device: (%0x) returned from CreateFile\n", Status);
    return;
}

//Mandatory initialization part
SIO2_Out(0x2A, 0xFD); //Enable GPIO Port1 - IO0-IO8 digital IO
SIO2_Out(0x29, 0xFC); //Enable GPIO Port3 - IO9-IO15 digital IO
SIO2_Out(0x2B, 0xFF); //Enable GPIO Port2 - IO16-IO21 digital IO

// IO0-IO7 belongs to Device 7.
// IO8-IO15 - Device 8
// IO16-IO21 - Device 9
// Devices have to be activated
SIO2Dev_Out(7, 0x30, 1); //Activate GPIO Port1
SIO2Dev_Out(8, 0x30, 1); //Activate GPIO Port2
SIO2Dev_Out(9, 0x30, 1); //Activate GPIO Port3
//End of mandatory initialization part

//Assignment of input or output - Addrees 0xF0 in all 3 devices
//write "1" to set pin as digital input, "0" to set pin as digital output
SIO2Dev_Out(7, 0xF0, 0xFF); //IO0-IO7 assigned as input
SIO2Dev_Out(8, 0xF0, 0x00); //IO8-IO15 assigned as output
SIO2Dev_Out(9, 0xF0, 0x07); //IO16-IO18 - inputs, IO19-IO21 - outputs

//Use inverter option for input and output - Address 0xF2 in all 3 devices
//write "1" to enable inverter, "0" - to disable.
SIO2Dev_Out(7, 0xF2, 0x00); //IO0-IO7 - not inverted
SIO2Dev_Out(8, 0xF2, 0x00); //IO8-IO15 - not inverted
SIO2Dev_Out(9, 0xF2, 0x00); //IO16-IO21 - not inverted

//Access to digital I/O
//Address 0xF1 for all 3 devices
SIO2Dev_Out(8, 0xF1, 0x55); //output 0x55 value to digital output IO8-IO15
unsigned char b = SIO2Dev_In(7, 0xF1); //8-bits input of IO0-IO7

Close_Int15dl(); //Close Int15 driver
```

Registers structure (file Int15srv.h):

It's used for exchanging information between user program and "Int15dl" driver in EEPROM, Temperature and WatchDOG.

```
typedef struct _Registers {
    union {
        struct {
            unsigned short ax;
            unsigned short bx;
            unsigned short cx;
            unsigned short dx;
            unsigned short bp;
            unsigned short si;
            unsigned short di;
            unsigned short ds;
            unsigned short es;
            unsigned short flags;
        };
        struct {
            unsigned char al;
            unsigned char ah;
            unsigned char bl;
            unsigned char bh;
            unsigned char cl;
            unsigned char ch;
            unsigned char dl;
            unsigned char dh;
        };
    };
} TRegisters;
```