Общество с ограниченной ответственностью

«Модульные Системы Торнадо»

Программное обеспечение

Программа «MIRage-N-microcode» для устройств ввода/вывода серии MIRage-N

Текст программы (исходный код)

АБНС.53011-01 12 01

Технический директор С.А. Кулагин

2020

Согласовано	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

```
-----------------------------------------------------------------------
```

Start12.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <start12.h>
#include <types.h>


/* Macros to control how the startup code handles the COP: */
/* #define _DO_FEED_COP_  : do feed the COP  */
/* #define _DO_ENABLE_COP_: do enable the COP  */
/* #define _DO_DISABLE_COP_: disable the COP */
/* Without defining any of these, the startup code does NOT handle the COP */

#pragma DATA_SEG __NEAR_SEG STARTUP_DATA /* _startupData can be accessed using 
16 bit accesses. This is needed because it contains the stack top, and without stack, 
far data cannot be accessed */
struct _tagStartup _startupData;  /*   read-only: */
                                  /*   _startupData is allocated in ROM and */
                                  /*   initialized by the linker */
#pragma DATA_SEG DEFAULT

#include "default.sgm"

static void Init(void) {
/* purpose:     1) zero out RAM-areas where data is allocated   */
/*              2) copy initialization data from ROM to RAM     */
/*              3) call global constructors in C++              */
/*   called from: _Startup, LibInits                            */
   __asm {
ZeroOut:
             LDX   _startupData.pZeroOut    ; *pZeroOut
             LDY   _startupData.nofZeroOuts ; nofZeroOuts
             BEQ   CopyDown                 ; if nothing to zero out
NextZeroOut: PSHY                           ; save nofZeroOuts
             LDY    2,X+                           ; start address and advance 
*pZeroOut (X = X+4)
             LDD   2,X+                     ; byte count
NextWord:    CLR   1,Y+                     ; clear memory byte
             DBNE  D, NextWord              ; dec byte count
             PULY                           ; restore nofZeroOuts
             DEY                            ; dec nofZeroOuts
             BNE  NextZeroOut
CopyDown:
             LDX   _startupData.toCopyDownBeg ; load address of copy down desc.
NextBlock:
             LDD   2,X+                     ; size of init-data -> D
             BEQ   funcInits                ; end of copy down desc.
             LDY   2,X+                     ; load destination address
Copy:        MOVB  1,X+,1,Y+                 ; move a byte from ROM to the data 
area
             DBNE  D,Copy                   ; copy-byte loop
             BRA   NextBlock
```

```
      funcInits:                                ; call of global construtors is on-
ly in c++ necessary
         }
      }

      #include "non_bank.sgm"

      #pragma  MESSAGE  DISABLE  C12053  /*  Stack-pointer  change  not  in  debugging-
information */
      #pragma NO_FRAME
      #pragma NO_ENTRY
      #pragma NO_EXIT

      void _Startup1(void) {
      /*  purpose:    1)  initialize the stack
                      2)  initialize the RAM, copy down init data etc (Init)
                      3)  call main;
          parameters: NONE
          called from: _PRESTART-code generated by the Linker
                       or directly referenced by the reset vector */
        for(;;) { /* forever: initialize the program; call the root-procedure */
          __asm LDS  _startupData.stackOffset
          Init(); /* zero out, copy down, call constructors */
          (*_startupData.main)();
          } /* end loop forever */
      }


      #pragma DATA_SEG NOINIT_RAM
      uint8 SystemResetSource;
      #pragma DATA_SEG DEFAULT


      #pragma NO_ENTRY
      #pragma NO_EXIT
      void __interrupt 0 _Startup(void) {
        __asm movb #0, SystemResetSource
        __asm jmp _Startup1
      }
      #pragma NO_ENTRY
      #pragma NO_EXIT
      void __interrupt 1 _StartupClkMon(void) {
        __asm movb #1, SystemResetSource
        __asm jmp _Startup1
      }
      #pragma NO_ENTRY
      #pragma NO_EXIT
      void __interrupt 2 _StartupCop(void) {
        __asm movb #2, SystemResetSource
        __asm jmp _Startup1
      }
      #pragma NO_ENTRY
      #pragma NO_EXIT
      void __interrupt 3 _StartupTrap(void) {
        __asm movb #3, SystemResetSource
        __asm jmp _Startup1
      }
```

Security.asm

```
KEY1:   EQU     0207
KEY2:   EQU     2009
KEY3:   EQU     2243
KEY4:   EQU     4321


        ORG     $FF00
        DC.W    KEY1, KEY2, KEY3, KEY4


codeSec: SECTION
        XDEF    Security

Security:
        leas    -(EndCode-StartCode),sp
        tfr     sp, y
        ldx     #StartCode
        ldd     #(EndCode-StartCode)
L0:     movb    1,x+, 1,y+
        dbne    d, L0
        tfr     sp, x
        jsr     0,x
        leas    (EndCode-StartCode),sp
        rts

StartCode:
        bset    $103, #$20
        movw    #KEY1, $FF00
        movw    #KEY2, $FF02
        movw    #KEY3, $FF04
        movw    #KEY4, $FF06
        bclr    $103, #$20
        rts

EndCode:
        end
```

Hw_defs.h

```
#ifndef _HW_DEFS_H_
#define _HW_DEFS_H_

#define OSCCLK            16000000


#define LED_ON_PU           0x02
#define LED_ON_PIN          PTU_PTU1
#define PWM_PU              0x08
#define PWM_PIN             PTU_PTU3
#define PFAULT_DISABLE_PU   0x40
#define PFAULT_DISABLE_PIN  PTU_PTU6
#define PWM_DISABLE_PU      0x80
#define PWM_DISABLE_PIN     PTU_PTU7
#define KEY_PT              0x01
```

```
#define KEY_PIN             PTT_PTT0
#define EEP_CS_PS           0x80
#define EEP_CS_PIN          PTS_PTS7

/*EEprom on IO*/
#define DDRPEEIO            DDRK
#define PEEIO               PORTK
#define EEPIO_CS_PEEIO      0x80
#define EEPIO_CS_PIN        PORTK_BIT7

/* io-dependet */
#define LMIO73_CLK_PK       0x40
#define LMIO73_CLK_PIN      PORTK_BIT6
#define LMIO73_DAT_PK       0x20
#define LMIO73_DAT_PIN      PORTK_BIT5



#endif
```

Sysinit.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "hw_defs.h"

void Security(void);


void SysInit() {
  /*system init*/
  DisableInterrupts;
  INTCR_IRQEN = 0;
  Security();
  /*external bus configuration*/
  MODE = 0xA0;    //Normal Expanded Narrow Mode
  DDRE |= 0x10;   //PE4(ECLK) as output
  PEAR = 0x14;      //PE4(ECLK) is a general-purpose I/O pin, PE2 is configured
as the R/W pin
  EBICTL = 0x01;  //ECLK stretches high during stretched external accesses
  MISC = 0x05;    //ECLK Stretched = 1; ROM On
  PEAR &= ~0x10;  //PE4(ECLK) is the external E clock pin
  /*clock and PLL*/
#ifndef _SIMULATOR_
  /*System clocks from PLLCLK=16*2*(SYNR+1)/(REFDV+1)MHz, Bus Clock = PLLCLK/2
= 16MHz*/
  REFDV = 0;
  SYNR = 0;
  /*System clocks from PLLCLK=16*2*(SYNR+1)/(REFDV+1)MHz, Bus Clock = PLLCLK/2
= 25MHz*/
//  REFDV = 15;
//  SYNR = 24;
  while(!(CRGFLG & 0x08)); //wait lock
  CLKSEL |= 0x80;
#endif //_SIMULATOR_
```

```
  if(DBGC1_DBGEN == 1) //if special mode
    COPCTL = 0x40;  //Stops the COP and RTI counters in active BDM mode
  /*port init to default: pull-ups*/
  PTU = PFAULT_DISABLE_PU | PWM_DISABLE_PU;
  DDRU = LED_ON_PU | PWM_PU | PFAULT_DISABLE_PU | PWM_DISABLE_PU;
  PERU = 0x31;
  PERM = 0xFB;
  PERQ = 0x7F;
  PERT = 0xFF;
  PERS = 0xFF;
  PERAD = 0xFFFF;
  ATDDIEN = 0xFFFF;
  PERP = 0x3F;
  PUCR = 0x93;  //added 5.08.2011
}
```

Main.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "variable.h"
#include "sysinit.h"
#include "eeprom.h"
#include "ident.h"
#include "mode.h"
#include "net.h"
#include "rtc.h"
#include "lm73.h"


extern uint8 SystemResetSource;


void main(void) {
  ResetStatus = SystemResetSource;
  SysInit();
  EE2Init();
  EEInit(0);
  if(Identification() != 0) for(;;);  //forever loop
  RTISetup(DevProfile->rtr, DevProfile->proc_time_ms);
  NETInit();
  EnableInterrupts;
  if(ModeSetup() == SPEC_MODE_SETUP_IP)
    NETSetupIP();
  NETInitializeIP(1);
  LM73Init();
  MainInit();
  if(DBGC1_DBGEN == 0) {
    ARMCOP = 0x55; ARMCOP = 0xAA;
    COPCTL = 0x47; //enable COP with timeout ~1sec
    }
  MainLoop();
}
```

Ident.h

```
#ifndef _IDENT_H_
#define _IDENT_H_
```

```
typedef struct {
  struct {
    uint8 len;
    char const *str;
    } rec[3];
} typeDevName;


typedef struct {
  typeDevName DevName;
  uint8 rtr;
  uint16 proc_time_ms;
  uint8 ppage;
  void (*IntXIRQHandlerProc)(void);
  void (*IntIRQHandlerProc)(void);
  void (*IntRTIHandlerProc)(void);
  void (*IntKWUHandlerProc)(void);
  void (*MainInit)(void);
  void (*MainLoop)(void);
} typeDevProfile;

#ifndef _OWN_HEADER_
extern typeDevProfile *DevProfile;
extern void (*IntXIRQHandlerProc)(void);
extern void (*IntIRQHandlerProc)(void);
extern void (*IntRTIHandlerProc)(void);
extern void (*IntKWUHandlerProc)(void);
extern void (*MainInit)(void);
extern void (*MainLoop)(void);
#endif

int Identification(void);


#endif


----------------------------------------------------------------------
```
Ident.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include <string.h>
#include "net.h"
#include "eeprom.h"
#include "variable.h"
#define  _OWN_HEADER_
#include "ident.h"


/***************** Declaration *********************/

const char _Company_[] = "Tornado";

const char _NXXX_Name_[] = "NXXX";
const char _NXXX_Version_[] = "V14.1.0";
void NXXX_IntIRQHandlerProc();
void NXXX_IntRTIHandlerProc();
void NXXX_MainInit();
void NXXX_MainLoop();
const typeDevProfile NXXX_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NXXX_Name_),_NXXX_Name_,
   sizeof(_NXXX_Version_),_NXXX_Version_},
  0x59,
  0,
  0x3D,
  NULL,
  NXXX_IntIRQHandlerProc,
  NXXX_IntRTIHandlerProc,
  NULL,
  NXXX_MainInit,
  NXXX_MainLoop,
};


const char _NPT_Name_[] = "NPT";
const char _NPT_Version_[] = "V14.1.3.5";
void NPT_IntIRQHandlerProc();
void NPT_MainInit();
void NPT_MainLoop();
const typeDevProfile NPT_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NPT_Name_),_NPT_Name_,
   sizeof(_NPT_Version_),_NPT_Version_},
  0x59,
  0,
  0x3D,
  NULL,
  NPT_IntIRQHandlerProc,
  NULL,
  NULL,
  NPT_MainInit,
  NPT_MainLoop,
};


const char _NTERM_Name_[] = "NTERM";
const char _NTERM_Version_[] = "V14.1.3.4";
void NTERM_IntIRQHandlerProc();
void NTERM_MainInit();
void NTERM_MainLoop();
const typeDevProfile NTERM_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NTERM_Name_),_NTERM_Name_,
   sizeof(_NTERM_Version_),_NTERM_Version_},
  0x59,
  0,
  0x3D,
  NULL,
  NTERM_IntIRQHandlerProc,
  NULL,
  NULL,
  NTERM_MainInit,
  NTERM_MainLoop,
};


const char _NAI_Name_[] = "NFAI";
const char _NAI_Version_[] = "V14.1.1.1";
void NAI_IntIRQHandlerProc();
void NAI_MainInit();
void NAI_MainLoop();
const typeDevProfile NAI_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NAI_Name_),_NAI_Name_,
   sizeof(_NAI_Version_),_NAI_Version_},
  0x59,
  0,
  0x3D,
  NULL,
  NAI_IntIRQHandlerProc,
  NULL,
  NULL,
  NAI_MainInit,
  NAI_MainLoop,
};


const char _NAO_Name_[] = "NAO";
const char _NAO_Version_[] = "V14.2.0.3";
void NAO_IntRTIHandlerProc();
void NAO_MainInit();
void NAO_MainLoop();
const typeDevProfile NAO_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NAO_Name_),_NAO_Name_,
   sizeof(_NAO_Version_),_NAO_Version_},
  0x60,
  0,
  0x3D,
  NULL,
  NULL,
  NAO_IntRTIHandlerProc,
  NULL,
  NAO_MainInit,
  NAO_MainLoop,
};


const char _NDIO_Name_[] = "NDIO";
const char _NDIO_Version_[] = "V14.3.1";
void NDIO_IntRTIHandlerProc();
void NDIO_MainInit();
void NDIO_MainLoop();
const typeDevProfile NDIO_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NDIO_Name_),_NDIO_Name_,
   sizeof(_NDIO_Version_),_NDIO_Version_},
  0x60,  //devider=32768 => interval=2.048 msec
  0,
  0x3D,
  NULL,
  NULL,
  NDIO_IntRTIHandlerProc,
  NULL,
  NDIO_MainInit,
  NDIO_MainLoop,
};


const char _N485_Name_[] = "N485";
const char _N485_Version_[] = "V14.10.1";
void N485_IntKWUHandlerProc();
void N485_MainInit();
void N485_MainLoop();
const typeDevProfile N485_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_N485_Name_),_N485_Name_,
   sizeof(_N485_Version_),_N485_Version_},
  0x59,
  0,
  0x3D,
  NULL,
  NULL,
  NULL,
  N485_IntKWUHandlerProc,
  N485_MainInit,
  N485_MainLoop,
};


const char _NAO4I_Name_[] = "NAO4I";
const char _NAO4I_Version_[] = "V14.0.1.1";
void NAO4I_IntRTIHandlerProc();
void NAO4I_MainInit();
void NAO4I_MainLoop();
const typeDevProfile NAO4I_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NAO4I_Name_),_NAO4I_Name_,
   sizeof(_NAO4I_Version_),_NAO4I_Version_},
  0x60,
  0,
  0x3D,
  NULL,
  NULL,
  NAO4I_IntRTIHandlerProc,
  NULL,
  NAO4I_MainInit,
  NAO4I_MainLoop,
};


const char _NTMU_Name_[] = "NTMU";
const char _NTMU_Version_[] = "V14.1.0.1";
void NTMU_IntXIRQHandlerProc();
void NTMU_IntIRQHandlerProc();
void NTMU_IntRTIHandlerProc();
void NTMU_MainInit();
void NTMU_MainLoop();
const typeDevProfile NTMU_DevProfile = {
  {sizeof(_Company_),_Company_,
   sizeof(_NTMU_Name_),_NTMU_Name_,
   sizeof(_NTMU_Version_),_NTMU_Version_},
  0x59,        //devider=163840 => interval=10.24 msec
  0,
  0x3D,
  NULL,
  NTMU_IntIRQHandlerProc,
  NTMU_IntRTIHandlerProc,
  NULL,
  NTMU_MainInit,
  NTMU_MainLoop,
};



const typeDevProfile *DevProfileList[] = {
  &NXXX_DevProfile, &NPT_DevProfile, &NTERM_DevProfile,
  &NAI_DevProfile, &NAO_DevProfile, &NDIO_DevProfile,
  &N485_DevProfile, &NAO4I_DevProfile, &NTMU_DevProfile, NULL
};
```

```
/****************** procedure *********************/

typeDevProfile const *DevProfile;
void (*IntXIRQHandlerProc)(void);
void (*IntIRQHandlerProc)(void);
void (*IntRTIHandlerProc)(void);
void (*IntKWUHandlerProc)(void);
void (*MainInit)(void);
void (*MainLoop)(void);


int Identification() {
  int i;
  char name[16];
  EE2_GET(DevName, name);
  for(i=0;;i++) {
    DevProfile = DevProfileList[i];
    if(DevProfile == NULL) return -1;
    if(strcmp(name, DevProfile->DevName.rec[1].str) == 0) {
      PPAGE = DevProfile->ppage;
      IntXIRQHandlerProc = DevProfile->IntXIRQHandlerProc;
      IntIRQHandlerProc = DevProfile->IntIRQHandlerProc;
      IntRTIHandlerProc = DevProfile->IntRTIHandlerProc;
      IntKWUHandlerProc = DevProfile->IntKWUHandlerProc;
      MainInit = DevProfile->MainInit;
      MainLoop = DevProfile->MainLoop;
      break;
      }
    }
  return 0;
}


-------------------------------------------------------------------
```

Variable.h

```
#ifndef _VARIABLE_H_
#define _VARIABLE_H_

#include <types.h>
#include "nxxx_types.h"
#include "npt_types.h"
#include "nterm_types.h"
#include "nai_types.h"
#include "nao_types.h"
#include "ndio_types.h"
#include "n485_types.h"
#include "nao4i_types.h"
#include "ntmu_types.h"


#ifndef _IMPLEMENTATION_
extern
#endif
union {
  struct {
    typeNXXX_Data Data;
    } NXXX_Var;
  struct {
    typeNPT_Data Data;
    typeNPT_EEIOData EEIOData;
    } NPT_Var;
  struct {
    typeNTERM_Data Data;
    typeNTERM_EE2HdData EE2HdData;
    typeNTERM_EEIOData EEIOData;
    } NTERM_Var;
  struct {
    typeNAI_Data Data;
    typeNAI_EEIOData EEIOData;
    int InitFlg;
    int32 AvrgBuff[4][16];
    uint16 AvrgPtr[4];
    uint32 ADCAccum;
    uint8 ADCCount, ADCCount00;
    } NAI_Var;
  struct {
    typeNAO_Data Data;
    typeNAO_EEIOData EEIOData;
    int16 DACValueCurrent[4];
    int16 AOValueCurrent[4];
    uint16 DIOCtrlPrev[8];
    uint8 DIOCtrlBitsPrev;
    } NAO_Var;
  struct {
    typeNDIO_Data Data;
    typeNDIO_Data2 Data2;
    uint32 CtrlBitsPrev;
    uint8 FiltrCount[32];
    uint8 FiltrLevel[32];
    uint8 FiltrTimeFactor1;
    uint8 FiltrTimeFactor2;
    } NDIO_Var;
  struct {
    typeN485_Data Data;
    uint8 *RcvBuff[4];
    uint16 RcvBuffPtr[4];
    uint8 SendBuff[4][365];
    uint16 SendBuffPtr[4];
    uint8 SendToIPFlg[4];
    uint8 RcvFlg[4];
    uint16 RcvBuffLeng[4];
    uint8 MBHostIC[4];
    uint32 MBHostIP[4];
    uint16 MBHostPort[4];
    uint8 WaitRstFlg;
    uint8 AVRMON_HushLooker[4];  //счетчик отсутствия приема от 485
    uint8 AVRMON_Fault[4];  //устанавливается для отложенного сброса
    uint8 AVRMON_Break[4];  //устанавливается для немедленного сброса
    } N485_Var;
  struct {
    typeNAO4I_Data Data;
    typeNAO4I_EEIOData EEIOData;
    int16 DACValueCurrent[4];
    int16 AOValueCurrent[4];
    int16 AOTypeCurrent[4];
    uint8 RTICount;
    } NAO4I_Var;
  struct {
    typeNTMU_Data Data;
    uint16 FPGAHalfPageCount;
    uint8 ReinitFlg;
    uint16 W51TrPtr;
    uint16 TxCount;
    uint8 ReadyFlg;
    uint16 SyncPrescalerPrev;
    } NTMU_Var;
```

```
} Vars;


#ifndef _IMPLEMENTATION_
extern
#endif
uint16 DataPrmCheckSum;

#ifndef _IMPLEMENTATION_
extern
#endif
uint16 EEIODataPrmCheckSum;

#ifndef _IMPLEMENTATION_
extern
#endif
uint16 EE2HdDataPrmCheckSum;

#ifndef _IMPLEMENTATION_
extern
#endif
uint8 ProcHandledFlg;

#ifndef _IMPLEMENTATION_
extern
#endif
uint8 ProtectKey;

#ifndef _IMPLEMENTATION_
extern
#endif
uint8 ResetStatus;


#endif



----------------------------------------------------------------------
```

Mode.h

```
#ifndef _HW_DEFS_H_
#define _HW_DEFS_H_

#define OSCCLK            16000000


#define LED_ON_PU           0x02
#define LED_ON_PIN          PTU_PTU1
#define PWM_PU              0x08
#define PWM_PIN             PTU_PTU3
#define PFAULT_DISABLE_PU   0x40
#define PFAULT_DISABLE_PIN  PTU_PTU6
#define PWM_DISABLE_PU      0x80
#define PWM_DISABLE_PIN     PTU_PTU7
#define KEY_PT              0x01
#define KEY_PIN             PTT_PTT0
#define EEP_CS_PS           0x80
#define EEP_CS_PIN          PTS_PTS7

/*EEprom on IO*/
#define DDRPEEIO            DDRK
#define PEEIO               PORTK
#define EEPIO_CS_PEEIO      0x80
```

```
#define EEPIO_CS_PIN        PORTK_BIT7

/* io-dependet */
#define LMIO73_CLK_PK       0x40
#define LMIO73_CLK_PIN      PORTK_BIT6
#define LMIO73_DAT_PK       0x20
#define LMIO73_DAT_PIN      PORTK_BIT5



#endif
```

Mode.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "hw_defs.h"
#include "rtc.h"
#define  _OWN_HEADER_
#include "mode.h"


uint8 SpecMode;
typeModeIndicationData IndicationData;
uint16 IndicatorTickCount0;
uint16 IndicatorPauseCount0;


uint8 ModeSetup() {
  uint16 cnt;
  uint8 k1, k, mode;
  uint16 set_specmode_time_count0;
  if(KEY_PIN) return 0;
  SpecMode = 0;
  set_specmode_time_count0 = RTIMSecToCount(3000);
  IndicatorTickCount0 = RTIMSecToCount(200);
  IndicatorPauseCount0 = RTIMSecToCount(1000);
  while(KEY_PIN == 0);
  mode = 1;
  LED_ON_PIN = 1;
  RTICommonTimerCount = set_specmode_time_count0;
  cnt = 0; k = 0; k1 = 0;
  while(RTICommonTimerCount) {
    if(KEY_PIN && (cnt<20000)) cnt++;
    if(!KEY_PIN) {
      RTICommonTimerCount = set_specmode_time_count0;
      if(cnt) cnt--;
      }
    if(cnt == 0) k = 0;
    if(cnt == 20000) k = 1;
    if((k1==1)&&(k==0)) mode++;
    k1 = k;
    }
  LED_ON_PIN = 0;
  SpecMode = mode;
  return mode;
}


void ModeIntRTIHandlerProc() {
  if(IndicationData.Count == 0) {
    IndicationData.Count = IndicatorTickCount0;
    if(IndicationData.PauseCount == 0) {
      if(IndicationData.BlinkCount) {
        LED_ON_PIN ^= 1;
        IndicationData.BlinkCount--;
        }
      else IndicationData.PauseCount = IndicatorPauseCount0;
      }
    }
  if(IndicationData.PauseCount) {
    if(--IndicationData.PauseCount == 0)
      IndicationData.BlinkCount = (SpecMode << 1);
    }
  IndicationData.Count--;
}
```

Irq.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "hw_defs.h"
#include "D:/NXXX/Soft/MC_LIB/lib/socklib51/sources/w5100io.h"
#include "D:/NXXX/Soft/MC_LIB/lib/socklib51/sources/socklib51.h"


extern void (*IntXIRQHandlerProc)(void);
extern void (*IntIRQHandlerProc)(void);
extern void (*IntKWUHandlerProc)(void);


void interrupt 5 _IntXIRQHandler() {
  if(IntXIRQHandlerProc) IntXIRQHandlerProc();
}


void interrupt 6 _IntIRQHandler() {
  if((PTIS & EXTINT_PS)==0) W51IRQHandler();
  else
    if(IntIRQHandlerProc) IntIRQHandlerProc();
}


void interrupt 24 _IntKWUHandler() {
  if(IntKWUHandlerProc) IntKWUHandlerProc();
}
```

Eeprom.h

```
#ifndef _EEPROM_H_
#define _EEPROM_H_


/******************** EE2 Layout define ********************/
#define EE2_SIZE  512

typedef struct {
  char DevName[16];     //0+16
  struct {
    uint8 val[6];
    } MAC[2];           //16+12
  uint32 IP[2];         //28+8
  uint32 SubMask[2];    //36+8
  uint32 Gateway[2];    //44+8
  uint8 res1[74];       //52+74
  uint16 PrmCheckSum;   //126+2
#ifdef typeEE2Prm
  typeEE2Prm Prm;       //128+...
  uint8 res2[126-sizeof(typeEE2Prm)];
#else
  uint8 res3[126];
#endif
  uint16 HdPrmCheckSum; //254+2
#ifdef typeEE2HdPrm
  typeEE2HdPrm HdPrm;   //256+..
#endif
} typeEE2Layout;


/******************** EEIO Layout define *******************/
#define EEIO_SIZE 2048

typedef struct {
  uint8 res1[126];
  uint16 PrmCheckSum;   //126+2
#ifdef typeEEIOPrm
  typeEEIOPrm Prm;
#endif
} typeEEIOLayout;


/**********************************************************/


#define EE_CS(x,y) {if(x) EEPIO_CS_PIN=y;else EEP_CS_PIN=y;}
//#define                                                     SPI_EXCHANGE(x,y)
{while(!(SPISR&0x20));SPIDR=x;while(!(SPISR&0x80));y=SPIDR;}

uint8 SPI_Exchange(uint8 data);
void SPI_MExchange(uint8 *data_in, uint8 *data_out, int len);
void EEInit(uint8 to_eeio);
void EEWrite(uint16 addr, uint8 *data, uint16 len, uint8 to_eeio);
void EERead(uint16 addr, uint8 *data, uint16 len, uint8 to_eeio);
uint16 EEIOWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len);
uint16 GetEEIOPrmCheckSum(uint16 len);
int I2C_Read(uint8 addr, uint8 *data, int count);
int I2C_Write(uint8 addr, uint8 *data, int count, int flg);
void EE2Init(void);
int EE2Write(uint16 addr, uint8 *data, uint16 len);
int EE2Read(uint16 addr, uint8 *data, uint16 len);
uint16 EE2PrmWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len);
uint16 EE2HdPrmWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len);
uint16 GetEE2PrmCheckSum(uint16 len);
uint16 GetEE2HdPrmCheckSum(uint16 len);

#define CHKSUM0   0xEEDA
#define EE2_GET(a, v) EE2Read(((uint16)&((typeEE2Layout*)0)->a), (uint8*)&(v),
sizeof(v))
#define EE2_SET(a, v) EE2Write(((uint16)&((typeEE2Layout*)0)->a), (uint8*)&(v),
sizeof(v))
#define EE2_SETCKU(a, v) EE2PrmWriteCheckUpdate(((uint16)&((typeEE2Layout*)0)-
>a), (uint8*)&(v), sizeof(v))
```

```
      #define                            EE2HD_SETCKU(a,                          v)
EE2HdPrmWriteCheckUpdate(((uint16)&((typeEE2Layout*)0)->a), (uint8*)&(v), sizeof(v))
      #define EEIO_GET(a, v) EERead(((uint16)&((typeEEIOLayout*)0)->a), (uint8*)&(v),
sizeof(v), 1)
      #define     EEIO_SET(a,     v)     EEWrite(((uint16)&((typeEEIOLayout*)0)->a),
(uint8*)&(v), sizeof(v), 1)
      #define EEIO_SETCKU(a, v) EEIOWriteCheckUpdate(((uint16)&((typeEEIOLayout*)0)-
>a), (uint8*)&(v), sizeof(v))


      #endif



      --------------------------------------------------------------------
```

Eeprom.c

```
      #include <hidef.h>
      #include <mc9s12e64.h>
      #include <types.h>
      #include "D:/NXXX/Soft/MC_LIB/lib/cmnlib/sources/cmnlib.h"
      #include "hw_defs.h"
      #include "eeprom.h"



      uint8 SPI_Exchange(uint8 data) {
        while((SPISR & 0x20)==0);
        SPIDR = data;
        while((SPISR & 0x80)==0);
        return SPIDR;
      }


      void SPI_MExchange(uint8 *data_in, uint8 *data_out, int len) {
        int i;
        if(data_in == NULL) {
          for(i=0; i<len; i++) SPI_Exchange(data_out[i]);
          }
        else if(data_out == NULL) {
          for(i=0; i<len; i++) data_in[i] = SPI_Exchange(0);
          }
        else {
          for(i=0; i<len; i++) data_in[i] = SPI_Exchange(data_out[i]);
          }
      }


      /*-------- EE ----------*/

      void EEInit(uint8 to_eeio) {
        /*SPI init*/
        SPIBR = 0x04;  //SPI baud 500kHz @16MHz, 781.25kHz @25MHz
        SPICR1 = 0x50;  //SPI enable; Master mode
        if(to_eeio) {
          DDRPEEIO |= EEPIO_CS_PEEIO;
          PEEIO |= EEPIO_CS_PEEIO;
          }
        else {
          DDRS |= EEP_CS_PS;
          PTS |= EEP_CS_PS;
          }
      }
```

```
void EEWrite(uint16 addr, uint8 *data, uint16 len, uint8 to_eeio) {
  uint8 spi_data;
  uint8 _ccr;
  while(len) {
    SAVE_CCR(_ccr); __asm sei;
    EE_CS(to_eeio, 0);
    SPI_Exchange(6); //write enable
    EE_CS(to_eeio, 1);
    RESTORE_CCR(_ccr);
    asm nop; asm nop;
    SAVE_CCR(_ccr); __asm sei;
    EE_CS(to_eeio, 0);
    SPI_Exchange(2);
    SPI_Exchange(addr >> 8);
    SPI_Exchange(addr & 0xFF);
    while(len) {
      SPI_Exchange(*data);
      data++; len--; addr++;
      if((addr & 0x1F)==0) break;
      }
    EE_CS(to_eeio, 1);
    RESTORE_CCR(_ccr);
    asm nop; asm nop;
    do {  //wait for ready
      SAVE_CCR(_ccr); __asm sei;
      EE_CS(to_eeio, 0);
      SPI_Exchange(5);  //read status register
      spi_data = SPI_Exchange(0);
      EE_CS(to_eeio, 1);
      RESTORE_CCR(_ccr);
      } while(spi_data & 1);
    }/*while(len)*/
}


void EERead(uint16 addr, uint8 *data, uint16 len, uint8 to_eeio) {
  uint8 _ccr;
  SAVE_CCR(_ccr); __asm sei;
  EE_CS(to_eeio, 0);
  SPI_Exchange(3);
  SPI_Exchange(addr >> 8);
  SPI_Exchange(addr & 0xFF);
  while(len) {
    *data = SPI_Exchange(0);
    data++; len--;
    }
  EE_CS(to_eeio, 1);
  RESTORE_CCR(_ccr);
}


uint16 EEIOWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len) {
  uint16 i, upd, chksum;
  uint8 d;
  for(i=0, chksum=0; i<len; i++) {
    EERead(addr+i, &d, 1, 1);
    chksum += (int16)(data[i] - d);
    }
  EEWrite(addr, data, len, 1);
  EEIO_GET(PrmCheckSum, upd);
  upd -= chksum;
  EEIO_SET(PrmCheckSum, upd);
  return upd;
```

```
}


uint16 GetEEIOPrmCheckSum(uint16 len) {
  int i;
  uint8 d;
  uint16 check_sum;
  for(i=0, check_sum=CHKSUM0; i<len; i++) {
    EERead(((uint16)&((typeEEIOLayout*)0)->PrmCheckSum)+2 + i, &d, 1, 1);
    check_sum += (uint16)d;
    }
  return check_sum;
}


/*-------- EE2 -----------*/

void EE2Init() {
  PTM_PTM7 = 0; DDRM_DDRM7 = 1;
  Delay16_ms(70);  //hold SCL low for reset LM73
  DDRM_DDRM7 = 0;
  IBFD = 0x58;  //100kHz @16MHz, 156.25kHz @25MHz SCL clock
  IBCR = IBCR_IBEN_MASK;  //I2C enable
}


int I2C_Read(uint8 addr, uint8 *data, int count) {
  int i, err;
  uint8 tmp;
  err = 0;
  if(count == 0) return 0;
  do {
    IBCR = IBCR_IBEN_MASK;
    while(IBSR_IBB);
    IBCR |= IBCR_MS_SL_MASK | IBCR_TX_RX_MASK;
    IBSR_IBIF = 1;  //clear interrupt flag;
    IBDR = addr | 0x01;  //send slave address
    while(IBSR_IBIF == 0);
    } while(IBSR_RXAK);
  IBCR_TX_RX = 0;  //to read mode
  if(count == 1) IBCR_TXAK = 1;  //no ack if last byte
  IBSR_IBIF = 1;
  tmp = IBDR;  //dummy read
  i = 0;
  for(;;) {
    while(IBSR_IBIF == 0);
    if((IBSR_RXAK == 0)||(IBCR_TXAK)) {
      count--;
      if(count == 1)
        IBCR_TXAK = 1;
      else if(count == 0) {
        IBCR_MS_SL = 0;  //stop
        data[i] = IBDR; asm nop;
        break;
        }
      IBSR_IBIF = 1;
      data[i++] = IBDR;
      }
    else {
      err = -1;
      break;
      }
    }
  if(err)
```

```
    IBCR = 0;
  else
    IBCR = IBCR_IBEN_MASK;
  return err;
}


int I2C_Write(uint8 addr, uint8 *data, int count, int flg) {
  int i, err;
  err = 0;
  if(flg & 0x01) {  //with start
    do {
      IBCR = IBCR_IBEN_MASK;
      while(IBSR_IBB);
      IBCR |= IBCR_MS_SL_MASK | IBCR_TX_RX_MASK;
      IBSR_IBIF = 1;
      IBDR = addr & ~0x01;  //send slave address
      while(IBSR_IBIF == 0);
      } while(IBSR_RXAK);
    }
  i = 0;
  for(;;) {
    IBSR_IBIF = 1;
    IBDR = data[i++];
    while(IBSR_IBIF == 0);
    if(IBSR_RXAK) {
      err = -1;
      break;
      }
    count--;
    if(count == 0) break;
    }
  if(err)
    IBCR = 0;
  else {
    if(flg & 0x02) {  //with stop
      IBCR = IBCR_IBEN_MASK;
      }
    }
  return err;
}


int EE2Write(uint16 addr, uint8 *data, uint16 len) {
  int i;
  uint8 i2caddr;
  for(i=0; i<len; i++) {
    i2caddr = 0xA0 | ((*((uint8*)&addr) & 0x07)<<1);
    if(I2C_Write(i2caddr, (uint8*)&addr+1, 1, 0x01)) return -1;
    if(I2C_Write(i2caddr, &data[i], 1, 0x02)) return -1;
    addr++;
    }
  return 0;
}


int EE2Read(uint16 addr, uint8 *data, uint16 len) {
  int i;
  uint8 i2caddr;
  for(i=0; i<len; i++) {
    i2caddr = 0xA0 | ((*((uint8*)&addr) & 0x07)<<1);
    if(I2C_Write(i2caddr, (uint8*)&addr+1, 1, 0x03)) return -1;
    if(I2C_Read(i2caddr, &data[i], 1)) return -1;
    addr++;
    }
  return 0;
}


uint16 EE2PrmWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len) {
  uint16 i, upd, chksum;
  uint8 d;
  for(i=0, chksum=0; i<len; i++) {
    EE2Read(addr+i, &d, 1);
    chksum += (int16)(data[i] - d);
    }
  EE2Write(addr, data, len);
  EE2_GET(PrmCheckSum, upd);
  upd -= chksum;
  EE2_SET(PrmCheckSum, upd);
  return upd;
}


uint16 EE2HdPrmWriteCheckUpdate(uint16 addr, uint8 *data, uint16 len) {
  uint16 i, upd, chksum;
  uint8 d;
  for(i=0, chksum=0; i<len; i++) {
    EE2Read(addr+i, &d, 1);
    chksum += (int16)(data[i] - d);
    }
  EE2Write(addr, data, len);
  EE2_GET(HdPrmCheckSum, upd);
  upd -= chksum;
  EE2_SET(HdPrmCheckSum, upd);
  return upd;
}


uint16 GetEE2PrmCheckSum(uint16 len) {
  int i;
  uint8 d;
  uint16 check_sum;
  for(i=0, check_sum=CHKSUM0; i<len; i++) {
    EE2Read(((uint16)&((typeEE2Layout*)0)->PrmCheckSum)+2 + i, &d, 1);
    check_sum += (uint16)d;
    }
  return check_sum;
}


uint16 GetEE2HdPrmCheckSum(uint16 len) {
  int i;
  uint8 d;
  uint16 check_sum;
  for(i=0, check_sum=CHKSUM0; i<len; i++) {
    EE2Read(((uint16)&((typeEE2Layout*)0)->HdPrmCheckSum)+2 + i, &d, 1);
    check_sum += (uint16)d;
    }
  return check_sum;
}
```

Lm73.c

```
#include <hidef.h>
```

```
#include <mc9s12e64.h>
#include <types.h>
#include "D:/NXXX/Soft/MC_LIB/lib/cmnlib/sources/cmnlib.h"
#include "hw_defs.h"
#include "eeprom.h"
#include "lm73.h"


/*-------- LM73 at MC -----------*/

void LM73Init() {
  uint16 dd;
  dd = 0x01E0;
  I2C_Write(0x92, (uint8*)&dd, 2, 0x03);  //write configuration with Power Down
  Delay16_ms(100);
  dd = 0x0160;
  I2C_Write(0x92, (uint8*)&dd, 2, 0x03);  //write configuration with Power Up
}


int16 LM73GetTemperature() {
  int16 dd, t;
  dd = 0;
  I2C_Write(0x92, (uint8*)&dd, 1, 0x03);
  I2C_Read(0x92, (uint8*)&dd, 2);
  t = (int16)(*((int8*)&dd));
  t <<= 1;
  if(dd & 0x80) t |= 1;
  return t;
}


/*-------- LM73 at IO -----------*/

void _lmio73_clk_pulse() {
  Delay16_us(2);
  LMIO73_CLK_PIN = 1; Delay16_us(1);
  LMIO73_CLK_PIN = 0; Delay16_us(1);
}

void _lmio73_ack_by_lm() {
  DDRK &= ~LMIO73_DAT_PK;
  _lmio73_clk_pulse();
}

void _lmio73_ack_by_mc() {
  LMIO73_DAT_PIN = 0;
  DDRK |= LMIO73_DAT_PK;
  _lmio73_clk_pulse();
  DDRK &= ~LMIO73_DAT_PK;
}

void _lmio73_start_by_mc() {
  LMIO73_DAT_PIN = 0;
  DDRK |= LMIO73_DAT_PK;
  Delay16_us(2);
  LMIO73_CLK_PIN = 0;
  DDRK &= ~LMIO73_DAT_PK;
}

void _lmio73_stop_by_mc() {
  LMIO73_DAT_PIN = 0;
  DDRK |= LMIO73_DAT_PK;
  Delay16_us(2);
  LMIO73_CLK_PIN = 1;
  Delay16_us(2);
  LMIO73_DAT_PIN = 1;
  DDRK &= ~LMIO73_DAT_PK;
}

void _lmio73_send_data(uint8 shd) {
  int8 i;
  DDRK |= LMIO73_DAT_PK;
  for(i=0; i<8; i++) {
    if(shd & 0x80) LMIO73_DAT_PIN = 1;
    else LMIO73_DAT_PIN = 0;
    _lmio73_clk_pulse();
    shd <<= 1;
    }
  DDRK &= ~LMIO73_DAT_PK;
}

uint8 _lmio73_get_data() {
  int8 i, data;
  Delay16_us(2);
  for(i=0; i<8; i++) {
    data <<= 1;
    if(LMIO73_DAT_PIN) data |= 0x01;
    else data &= ~0x01;
    _lmio73_clk_pulse();
    }
  return data;
}

void _lmio73_write(uint8 addr, uint8 *data, int8 size) {
  int i;
  _lmio73_start_by_mc();
  _lmio73_send_data(addr);
  _lmio73_ack_by_lm();
  for(i=0; i<size; i++) {
    _lmio73_send_data(data[i]);
    _lmio73_ack_by_lm();
    }
  _lmio73_stop_by_mc();
}

void _lmio73_read(uint8 addr, uint8 *data, int8 size) {
  int i;
  _lmio73_start_by_mc();
  _lmio73_send_data(addr | 1);
  _lmio73_ack_by_lm();
  i = 0;
  for(;;) {
    if(i == size) break;
    data[i] = _lmio73_get_data();
    i++;
    if(i != size) _lmio73_ack_by_mc();
    else _lmio73_ack_by_lm();
    }
  _lmio73_stop_by_mc();
}


/*-----------------------------------------*/

void LMIO73Init() {
  uint16 dd;
  DDRK |= LMIO73_CLK_PK;
  LMIO73_CLK_PIN = 0;
  Delay16_ms(100);  //wait for reset I2C
  LMIO73_CLK_PIN = 1;
  dd = 0x01E0;
  _lmio73_write(0x92, (uint8*)&dd, 2);  //write configuration with Power Down
  Delay16_ms(100);
  dd = 0x0160;
  _lmio73_write(0x92, (uint8*)&dd, 2);  //write configuration with Power Up
}


int16 LMIO73GetTemperature() {
  /*~250us @25MHz execute time*/
  int16 dd, t;
  dd = 0;
  _lmio73_write(0x92, (uint8*)&dd, 1);
  _lmio73_read(0x92, (uint8*)&dd, 2);
  t = (int16)(*((int8*)&dd));
  t <<= 1;
  if(dd & 0x80) t |= 1;
  return t;
}
```

Net.h

```
#ifndef _NET_H_
#define _NET_H_

#define UDP_PORT_RX     502
#define UDP_PORT_TX     502
#define TCP_PORT        502

typedef struct {
  uint8 MACdst[6];
  uint8 MACsrc[6];
  uint16 Prot;
  struct {
    uint32 IPh0, IPh1, IPh2;
    uint32 IPsrc;
    uint32 IPdst;
    } hIP;
  struct {
    uint16 port_src;
    uint16 port_dst;
    uint16 leng;
    uint16 chksum;
    } hUDP;
  uint16 opcode;
  union {
    char name[1];
    struct {
      uint16 block;
      uint8 data[1];
      } tftp_rsp;
    } tftp;
} typeFrame;


/***  modbus over net  ***/

typedef struct {
  uint16 TransactionID;
  uint16 ProtocolID;
  uint16 Len;
} typeMBAP;

/*F2: Read Discrete Inputs*/
typedef struct {
  uint16 Addr;
  uint16 Count;
}typeF2Request;

typedef struct {
  uint8 Len;
  uint8 Data[1];
}typeF2Response;

/*F3: Read Holding Registers*/
typedef struct {
  uint16 Addr;
  uint16 Count;
} typeF3Request;

typedef struct {
  uint8 Len;
  uint16 Data[1];
} typeF3Response;

/*F6: Write Single Register*/
typedef struct {
  uint16 Addr;
  uint16 Data;
} typeF6Request;

typedef struct {
  uint16 Addr;
  uint16 Data;
} typeF6Response;

/*F16: Write Multiple registers*/
typedef struct {
  uint16 Addr;
  uint16 Count;
  uint8 Len;
  uint16 Data[1];
} typeF16Request;

typedef struct {
  uint16 Addr;
  uint16 Count;
} typeF16Response;

/*F21: Write file record*/
typedef struct {
  uint8 Len;
  uint8 RefType;
  uint16 FileNum;
  uint16 RecNum;
  uint16 RecLen;
  uint16 Data[1];
} typeF21Request;

typedef typeF21Request typeF21Response;
```

```
/*F23: *Write/Read Multiple registers*/
typedef struct {
  uint16 RdAddr;
  uint16 RdCount;
  uint16 WrAddr;
  uint16 WrCount;
  uint8 WrLen;
  uint16 Data[1];
} typeF23Request;

typedef struct {
  uint8 Len;
  uint16 Data[1];
} typeF23Response;

/*F43: Read Device Identification*/
typedef struct {
  uint8 MEIType;
  uint8 IDCode;
  uint8 ObjectID;
} typeF43Request;

typedef struct {
  uint8 header[6];
} typeF43Response;


typedef struct {
  typeMBAP MBAP;
  uint8 Slave;
  uint8 FCode;
  union {
    typeF2Request   F2Req;
    typeF2Response  F2Rsp;
    typeF3Request   F3Req;
    typeF3Response  F3Rsp;
    typeF6Request   F6Req;
    typeF6Response  F6Rsp;
    typeF16Request  F16Req;
    typeF16Response F16Rsp;
    typeF21Request  F21Req;
    typeF21Response F21Rsp;
    typeF23Request  F23Req;
    typeF23Response F23Rsp;
    typeF43Request  F43Req;
    typeF43Response F43Rsp;
    uint8           ExeptionCode;
    } ff;
} typeMB;

#define MBERR_ILLEGAL_FUNCTION        1
#define MBERR_ILLEGAL_DATA_ADDRESS    2
#define MBERR_ILLEGAL_DATA_VALUE      3

#ifndef _OWN_HEADER_
extern uint8 Buff[];
extern typeF43Response ObjectID;
#endif

void NETInit(void);
void NETSetupIP(void);
void NETInitializeIP(uint8 chk_give_flg);
void NETReinitSocket(uint8 ic, int8 s);
void NETTCPSendKeep(void);
```

```
int NETReadUDPPacket(uint8 ic, int8 s, uint8* buf, uint16 length);
int NETSend(uint8 ic, int8 s, uint8* buf, uint16 len);
int NETReceive(uint8 ic, int8 s);
void NETRTIHandlerProc(void);
uint8 NETSock_RxFlag(uint8 *p_ic, int8 *p_s);


#endif



-------------------------------------------------------------------
```

Net.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <ctype.h>
#include <stdio.h>
#include <types.h>
#define _SOCKLIB_
#include "D:/NXXX/Soft/MC_LIB/lib/socklib51/sources/w5100io.h"
#include "D:/NXXX/Soft/MC_LIB/lib/socklib51/sources/socklib51.h"
#include "hw_defs.h"
#include "eeprom.h"
#include "rtc.h"
#include "mode.h"
#define _OWN_HEADER_
#include "net.h"


uint8 Buff[MTU];
typeRAWHeader RAWHeader;
typeUDPHeader UDPHeader;
uint32 UDPHostIP[2][2];
uint16 UDPHostTimer[2][2];
uint16 UDPHostTimeoutCount0;
uint16 ReadUDPPacketTimer;
uint16 ReadUDPPacketTimeoutCount0;
const typeF43Response ObjectID = {0x0E, 1, 1, 0, 0, 3};
int errno;


void NETInit() {
  W51Reset();
  UDPHostTimeoutCount0 = RTISecToCount(10);
  ReadUDPPacketTimeoutCount0 = RTISecToCount(10);
  INTCR_IRQEN = 1;
}


void NETSetupIP() {
  uint8 ic;
  int8 s;
  int n, m;
  uint32 tmp;
  uint32 ip, submask, gateway;
  typeMAC mac[2], macf;
  typeFrame *frame;
  uint16 val[4];
  macf.val[0]=macf.val[1]=macf.val[2]=macf.val[3]=macf.val[4]=macf.val[5]     =
0xFF;
  tmp = 0;
  EE2_GET(MAC[0], mac[0]); Sock_SysInit(0, &mac[0], &tmp, &tmp, &tmp);
  EE2_GET(MAC[1], mac[1]); Sock_SysInit(1, &mac[1], &tmp, &tmp, &tmp);
```

```
        Sock_Init(0, 0, PROTOCOL_MACRAW | SK_MR_MF_WSK, 0);
        Sock_Init(1, 0, PROTOCOL_MACRAW | SK_MR_MF_WSK, 0);
        for(;;) {
          if(Sock_RxFlag(&ic, &s)) {
            Sock_Read(ic, s, (uint8*)&RAWHeader, sizeof(RAWHeader));
            NETReadUDPPacket(ic, s, Buff, RAWHeader.TLen-wo_sizeofRAWHeader);
            /*проверка на Broadcast*/
            frame = (typeFrame*)Buff;
            if((memcmp(frame->MACdst, &mac[ic], sizeof(typeMAC))==0) &&
               (memcmp(frame->MACdst, &macf, sizeof(typeMAC)) != 0)) {
              ip = frame->hIP.IPdst;
              EE2_SET(IP[ic], ip);
              m = 0;
              if(frame->opcode == 1) {
                for(n=0; (n<(frame->hUDP.leng - 4))&&(frame->tftp.name[n]); n++)
                  frame->tftp.name[n] = (char)tolower(frame->tftp.name[n]);
                if(sscanf(frame->tftp.name,  "mask=%d.%d.%d.%d",  &val[0],  &val[1],
&val[2], &val[3]) == 4)
                  m = 1;
                else  if(sscanf(frame->tftp.name,  "gateway=%d.%d.%d.%d",  &val[0],
&val[1], &val[2], &val[3]) == 4)
                  m = 2;
                if(m) {
                  *((uint8*)&tmp) = (uint8)val[0];
                  *((uint8*)&tmp+1) = (uint8)val[1];
                  *((uint8*)&tmp+2) = (uint8)val[2];
                  *((uint8*)&tmp+3) = (uint8)val[3];
                  if(m == 1) EE2_SET(SubMask[ic], tmp);
                  else EE2_SET(Gateway[ic], tmp);
                  }
                }
              EE2_GET(MAC[ic], mac[ic]);
              EE2_GET(IP[ic], ip);
              EE2_GET(SubMask[ic], submask);
              EE2_GET(Gateway[ic], gateway);
              Sock_SysInit(ic, &mac[ic], &ip, (uint32*)&submask, (uint32*)&gateway);
              Sock_Init(ic, s, PROTOCOL_UDP | SK_MR_MF_WSK, UDP_PORT_RX);
              if(m) {
                frame->opcode = 3;
                frame->tftp.tftp_rsp.block = 1;
                EE2Read(0, frame->tftp.tftp_rsp.data, 512);
                Sock_SendTo(ic,    s,    frame->hIP.IPsrc,    frame->hUDP.port_src,
(uint8*)&frame->opcode, 514, 0);
                Sock_SendTo(ic, s, 0, 0, NULL, 0, 0);
                tmp = 0;
                EE2_GET(MAC[ic],  mac[ic]);  Sock_SysInit(ic,  &mac[ic],  &tmp,  &tmp,
&tmp);
                Sock_Init(ic, s, PROTOCOL_MACRAW, 0);
                }
              }/*if(memcmp(frame->MACdst, &mac[ic], sizeof(typeMAC))==0)*/
            }
          if(SpecMode != SPEC_MODE_SETUP_IP) break;
          }
      }


      void NETInitializeIP(uint8 chk_give_flg) {
        uint8 ic;
        int8 i, j;
        uint32 ip, submask, gateway;
        typeMAC mac;
        for(ic=0; ic<2; ic++) {
          EE2_GET(MAC[ic], mac);
          EE2_GET(IP[ic], ip);
```

```
    EE2_GET(SubMask[ic], submask);
    EE2_GET(Gateway[ic], gateway);
    Sock_SysInit(ic, &mac, &ip, (uint32*)&submask, (uint32*)&gateway);
    Sock_Init(ic, 0, PROTOCOL_UDP | SK_MR_MF_WSK,  UDP_PORT_RX);
    Sock_Init(ic, 1, PROTOCOL_UDP | SK_MR_MF_WSK,  UDP_PORT_TX);
    Sock_Init(ic, 2, PROTOCOL_UDP | SK_MR_MF_WSK,  UDP_PORT_TX);
    Sock_Init(ic, 3, PROTOCOL_TCP | SK_MR_MF_WSK,  TCP_PORT);
    Sock_Listen(ic, 3);
    }
  for(i=0; i<2; i++) for(j=0; j<2; j++) UDPHostTimer[i][j] = 0;
  ReadUDPPacketTimer = 0;
  if(chk_give_flg) W51ChkGive();
}


void NETReinitSocket(uint8 ic, int8 s) {
  Sock_Close(ic, s);
  switch (s) {
    case 0:
      Sock_Init(ic, s, PROTOCOL_UDP | SK_MR_MF_WSK, UDP_PORT_RX);
    case 1:
    case 2:
      Sock_Init(ic, s, PROTOCOL_UDP | SK_MR_MF_WSK, UDP_PORT_TX);
      break;
    case 3:
      Sock_Init(ic, s, PROTOCOL_TCP | SK_MR_MF_WSK, TCP_PORT);
      Sock_Listen(ic, s);
      break;
    }
}


void NETTCPSendKeep() {
  Sock_TCPSendKeep(0, 3);
  Sock_TCPSendKeep(1, 3);
  }


int NETReadUDPPacket(uint8 ic, int8 s, uint8* buf, uint16 length) {
  uint16 rdlen;
  int len;
  rdlen = 0;
  ReadUDPPacketTimer = ReadUDPPacketTimeoutCount0;
  do {
    len = Sock_Read(ic, s, buf+rdlen, length - rdlen);
    if(len < 0) {
      NETReinitSocket(ic, s);
      return -1;
      }
    rdlen += (uint16)len;
    if(ReadUDPPacketTimer == 0) return -1; //if timeout
    } while(rdlen < length);
  return rdlen;
}


uint8 SelectUDPChannel(uint8 ic, int8 *p_s) {
  uint8 n, nn, reinit;
  uint32 *hostIP;
  uint16 *timer;
  reinit = 0;
  hostIP = UDPHostIP[ic];
  timer = UDPHostTimer[ic];
  for(n=0; n<2; n++) {
    if(hostIP[n] == UDPHeader.SIP) {
      if(timer[n] == 0) {
        nn = n ^ 1;
        if((hostIP[n] != hostIP[nn])&&(timer[nn] != 0)) {
          hostIP[n] = hostIP[nn];
          timer[n] = timer[nn];
          }
        else {
          reinit = 1;
          nn = 0;
          }
        }
      else {
        nn = n;
        }
      break;
      }
    }
  if(n == 2) {
    nn = (timer[0] > timer[1]) ? 1 : 0;
    }
  hostIP[nn] = UDPHeader.SIP;
  timer[nn] = UDPHostTimeoutCount0;
  *p_s = nn + 1;
  return reinit;
}


int NETSend(uint8 ic, int8 s, uint8* buf, uint16 len) {
  uint8 reinit_flg;
  switch(s) {
    case 0:
      reinit_flg = SelectUDPChannel(ic, &s);
      if(Sock_SendTo(ic,   s,   UDPHeader.SIP,   UDPHeader.SPort,   buf,   len,
reinit_flg) < 0) {
        NETReinitSocket(ic, s);
        return -1;
        }
      break;
    case 3:
      if(Sock_Send(ic, s, buf, len) < 0) {
        NETReinitSocket(ic, s);
        return -1;
        }
      break;
    }
  return 0;
}


int NETReceive(uint8 ic, int8 s) {
  switch (s) {
    case 0:
      if(Sock_Read(ic, s, (uint8*)&UDPHeader, sizeof(UDPHeader)) >= 0) {
        if(UDPHeader.TLen <= MTU) {
          return     NETReadUDPPacket(ic,     s,     Buff,     UDPHeader.TLen-
wo_sizeofUDPHeader);
          }
        else
          NETReinitSocket(ic, s);
        }
      break;
    case 3:
      return Sock_Read(ic, s, Buff, MTU);
    }
  return -1;
}


uint8 NETSock_RxFlag(uint8 *p_ic, int8 *p_s) {
  int8 chk, chkflg;
  chk = W51ChkCheck();
  if(chk) {
    NETInit();
    if(chk == ERR_CHK_CHECKSUM) chkflg = 1;
    else chkflg = 0;
    NETInitializeIP(chkflg);
    }
  return Sock_RxFlag(p_ic, p_s);
}


void NETRTIHandlerProc() {
  int n;
  for(n=0; n<4; n++)
    if(((uint16*)UDPHostTimer)[n]) ((uint16*)UDPHostTimer)[n]--;
  if(ReadUDPPacketTimer) ReadUDPPacketTimer--;
}
```

Rtc.h

```
#ifndef _RTC_H_
#define _RTC_H_

#ifndef _OWN_HEADER_
extern uint16 RTICommonTimerCount;
extern uint16 RTICommonTimerCount0;
#endif

void RTISetup(uint8 rtr, uint16 proc_time_ms);
uint16 RTISecToCount(uint16 sec);
uint16 RTIMSecToCount(uint16 msec);
void EnableRTIHandlerProc(int8 enable);


#endif
```

Rtc.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "hw_defs.h"
#include "net.h"
#include "mode.h"
#define  _OWN_HEADER_
#include "rtc.h"


extern void (*IntRTIHandlerProc)(void);
extern uint8 SpecMode;
uint16 RateHz;
uint16 IntRTIProcTimerCount;
```

```
uint16 IntRTIProcTimerCount0;
uint16 RTICommonTimerCount;
uint16 RTICommonTimerCount0;
int8 EnableRTIHandlerProcFlg;


void RTISetup(uint8 rtr, uint16 proc_time_ms) {
  RateHz                                                                 =
(uint16)(OSCCLK/((uint32)((rtr&0xF)+1)*((uint32)1<<(uint8)((rtr>>4)+9))));
  IntRTIProcTimerCount0 = RTIMSecToCount(proc_time_ms);
  IntRTIProcTimerCount = IntRTIProcTimerCount0;
  RTICommonTimerCount = 0;
  EnableRTIHandlerProcFlg = 0;
  RTICTL = rtr;
  CRGINT_RTIE = 1; //enable interrupt from RTC
}


uint16 RTISecToCount(uint16 sec) {
  return (sec * RateHz);
}


uint16 RTIMSecToCount(uint16 msec) {
  return (uint16)(((uint32)msec * RateHz)/1000);
}


void EnableRTIHandlerProc(int8 enable) {
  EnableRTIHandlerProcFlg = enable;
}


void interrupt 7 _IntRTIHandler() {
  NETRTIHandlerProc();
  if(SpecMode) ModeIntRTIHandlerProc();
  if(RTICommonTimerCount) RTICommonTimerCount--;
  if((IntRTIHandlerProc) && (EnableRTIHandlerProcFlg)) {
    if(IntRTIProcTimerCount) IntRTIProcTimerCount--;
    else {
      IntRTIProcTimerCount = IntRTIProcTimerCount0;
      IntRTIHandlerProc();
      }
    }
  CRGFLG_RTIF = 1;
}



-------------------------------------------------------------------
```

Utils.c

```
#include <hidef.h>
#include <types.h>
#include "eeprom.h"


int ParamCheck(uint8 *data, uint16 size, uint16 check_sum) {
  int i;
  uint16 sum;
  for(i=0, sum=CHKSUM0; i<size; i++) sum += (uint16)data[i];
  return (int)(sum + check_sum);
}
```

```
      int8 UpdateMBData(uint16 *mb_data, uint16 mb_addr, uint8 mb_count, void *data,
void *data_attr, uint8 key) {
        int8 flgs;
        uint16 _stx, _sty, _sty1;
        flgs = 0;
        __asm {
              stx     _stx
              sty     _sty
              ldd     mb_addr
              lsld
              tfr     d, y
              addd    data
              tfr     d, x        //x=&Data[mb_addr]
              tfr     y, d
              addd    data_attr
              addd    #1
              tfr     d, y        //y=&low(DataAttr[mb_addr])

      Lsd2:   tst     mb_count
              beq     Lsd_ret
              dec     mb_count

              ldab    2,y+        //attribute
              andb    key         //and with key mask
              tbeq    b, Lsd1
              orab    flgs
              stab    flgs        //flgs <= flgs | (attr & key)
              sty     _sty1
              ldy     mb_data
              movw    2,y+, 2,x+  //copy to data if (attr & key)!=0
              sty     mb_data
              ldy     _sty1
              bra     Lsd2
      Lsd1:   leax    2,x
              ldd     mb_data
              addd    #2
              std     mb_data
              bra     Lsd2
      Lsd_ret:
              ldy     _sty
              ldx     _stx
          }
        return flgs;
      }


      -------------------------------------------------------------------
```

Nxxx_type.h

```
      #ifndef _NXXX_TYPES_H_
      #define _NXXX_TYPES_H_


      //note: bit assign
      //      PortAD     : AD15...AD0
      //      PortSMPK   : -,-,PS6,PS5,PM5,PM4,PP1,PP0, PK7...PK0

      typedef struct {
        uint16 DataOutPortAD;       //0
        uint16 DataOutPortSMPK;                 //1
        uint16 DataInPortAD;                    //2
        uint16 DataInPortSMPK;                  //3
        uint16 DirPortAD;                                   //4
```

```
  uint16 DirPortSMPK;                           //5
  uint16 PurPortAD;           //6
  uint16 PurPortSMPK;         //7
  int16 res1[40];             //8..47
  uint16 RstStat;             //48
  int16 MCTmpr;               //49
  } typeNXXX_Data;


#define NXXX_DATA_ATTR \
   1,
                             /*DataOutPortAD*/   \
   1,
                             /*DataOutPortSMPK*/ \
   0,
                             /*DataInPortAD*/    \
   0,
                             /*DataInPortSMPK*/  \
   1,
                             /*DirPortAD*/       \
   1,
                             /*DirPortSMPK*/     \
   1,                                    /*PurPortAD*/       \
   1,
                             /*PurPortSMPK*/     \
   {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                             \
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                             \
    0,0,0,0,0,0,0,0},
                                                  \
   0,
                     /*RstStat*/                    \
   0
                             /*MCTmpr*/
/*Note: data_attr flag is:
   -----------------------------------------------------------------
   |   7   |   6   |   5   |   4   |   3   |   2   |   1   |   0   |
   |       |       |       |       |       |prot_ee| to_ee | wr_en |
   -----------------------------------------------------------------
*/


#endif



-------------------------------------------------------------------
```

Nxxx_init.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "variable.h"
#include "net.h"
#include "rtc.h"
#include "nxxx_defs.h"
#include "nxxx_proc.h"



void NXXX_MainInit() {
  SPICR1 = 0; //Reset SPI
  Vars.NXXX_Var.Data.RstStat = ResetStatus;
```

```
    Vars.NXXX_Var.Data.PurPortAD = 0xFFFF;
    Vars.NXXX_Var.Data.PurPortSMPK = 0xFFFF;
    RTICommonTimerCount = 0;
    nxxx_ProcInit();
    EnableRTIHandlerProc(1);
  }


  --------------------------------------------------------------------
```

Nxxx_loop.c

```
  #include <hidef.h>
  #include <mc9s12e64.h>
  #include <types.h>
  #include "D:/NXXX/Soft/MC_LIB/lib/cmnlib/sources/cmnlib.h"
  #include "D:/NXXX/Soft/MC_LIB/lib/socklib51/sources/socklib51.h"
  #include "net.h"
  #include "variable.h"
  #include "rtc.h"
  #include "lm73.h"
  #include "utils.h"
  #include "ident.h"
  #include "nxxx_defs.h"
  #include "nxxx_init.h"
  #include "nxxx_proc.h"


  const typeNXXX_Data NXXX_DataAttr = {NXXX_DATA_ATTR};


  void nxxx_MakeResponse(uint8 ic, int8 s, typeMB* MBpkt) {
    uint16 addr, leng, dd;
    int8 j, data_flgs;
    uint8 err;
    char _ccr;
    data_flgs = 0;
    err = 0;
    MBpkt->MBAP.Len = sizeof(typeMB)-sizeof(MBpkt->MBAP)-sizeof(MBpkt->ff);
    switch (MBpkt->FCode) {
      //--------------------------------------------------------------
      case 3: /*Read holding registers*/
        addr = MBpkt->ff.F3Req.Addr << 1;
        leng = MBpkt->ff.F3Req.Count << 1;
        if((addr+leng) <= sizeof(Vars.NXXX_Var.Data)) {
          NXXX_Proc();
          SAVE_CCR(_ccr); __asm sei;
          MemCpy(MBpkt->ff.F3Rsp.Data, (void*)((uint16)&Vars.NXXX_Var.Data+addr),
leng);
          RESTORE_CCR(_ccr);
          MBpkt->ff.F3Rsp.Len = (uint8)leng;
          MBpkt->MBAP.Len += leng + sizeof(MBpkt->ff.F3Rsp.Len);
          ProtectKey <<= 1; ProtectKey &= 0x04;
          }
        else {
          err = MBERR_ILLEGAL_DATA_ADDRESS;
          ProtectKey = 0;
          }
        break;
      //--------------------------------------------------------------
      case 6: /*Write Single Register*/
        if((MBpkt->ff.F6Req.Addr  +  1)  >  (sizeof(Vars.NXXX_Var.Data)>>1))  da-
ta_flgs = -1;
        else {
          SAVE_CCR(_ccr); __asm sei;
```

```
              data_flgs  =  UpdateMBData(&MBpkt->ff.F6Req.Data,  MBpkt->ff.F6Req.Addr,
1,
                                       &Vars.NXXX_Var.Data,    (void*)&NXXX_DataAttr,
0x03|ProtectKey);
              RESTORE_CCR(_ccr);
              NXXX_Proc();
              }
            if(data_flgs >= 0)
              MBpkt->MBAP.Len += sizeof(MBpkt->ff.F6Rsp);
            else {
              data_flgs = 0;
              err = MBERR_ILLEGAL_DATA_ADDRESS;
              }
            ProtectKey = 0;
            break;
          //-----------------------------------------------------------------
          case 16: /*Write Multiple registers*/
            if(MBpkt->ff.F16Req.Len == (MBpkt->ff.F16Req.Count << 1)) {
              if((MBpkt->ff.F16Req.Addr       +       MBpkt->ff.F16Req.Count)       >
(sizeof(Vars.NXXX_Var.Data)>>1)) data_flgs = -1;
              else {
                SAVE_CCR(_ccr); __asm sei;
                data_flgs      =      UpdateMBData(MBpkt->ff.F16Req.Data,      MBpkt-
>ff.F16Req.Addr, (uint8)MBpkt->ff.F16Req.Count,
                                         &Vars.NXXX_Var.Data,  (void*)&NXXX_DataAttr,
0x03|ProtectKey);
                RESTORE_CCR(_ccr);
                NXXX_Proc();
                }
              if(data_flgs >= 0)
                MBpkt->MBAP.Len += sizeof(MBpkt->ff.F16Rsp);
              else {
                data_flgs = 0;
                err = MBERR_ILLEGAL_DATA_ADDRESS;
                }
              }
            else
              err = MBERR_ILLEGAL_DATA_VALUE;
            ProtectKey = 0;
            break;
          //-----------------------------------------------------------------
          case 23: /*Write/Read Multiple registers*/
            if(MBpkt->ff.F23Req.WrLen == (MBpkt->ff.F23Req.WrCount << 1)) {
              addr = MBpkt->ff.F23Req.RdAddr << 1;
              leng = MBpkt->ff.F23Req.RdCount << 1;
              if((addr+leng) <= sizeof(Vars.NXXX_Var.Data)) {
                if((MBpkt->ff.F23Req.WrAddr     +     MBpkt->ff.F23Req.WrCount)     >
(sizeof(Vars.NXXX_Var.Data)>>1)) data_flgs = -1;
                else {
                  SAVE_CCR(_ccr); __asm sei;
                  data_flgs     =     UpdateMBData(MBpkt->ff.F23Req.Data,     MBpkt-
>ff.F23Req.WrAddr, (uint8)MBpkt->ff.F23Req.WrCount,
                                           &Vars.NXXX_Var.Data,
(void*)&NXXX_DataAttr, 0x03|ProtectKey);
                  RESTORE_CCR(_ccr);
                  NXXX_Proc();
                  }
                if(data_flgs >= 0) {
                                          SAVE_CCR(_ccr); __asm sei;
                  MemCpy(MBpkt->ff.F23Rsp.Data,
(void*)((uint16)&Vars.NXXX_Var.Data+addr), leng);
                  RESTORE_CCR(_ccr);
                  MBpkt->ff.F23Rsp.Len = (uint8)leng;
                  MBpkt->MBAP.Len += leng + sizeof(MBpkt->ff.F23Rsp.Len);
                  }
                else {
                  data_flgs = 0;
                  err = MBERR_ILLEGAL_DATA_ADDRESS;
                  }
                }
              else
                err = MBERR_ILLEGAL_DATA_ADDRESS;
              }
            else
              err = MBERR_ILLEGAL_DATA_VALUE;
            ProtectKey = 0;
            break;
          //---------------------------------------------------------------
          case 43: /*Basic device identification*/
            if((MBpkt->ff.F43Req.MEIType==0x0E)&&(MBpkt->ff.F43Req.IDCode==1)) {
              MemCpy(&MBpkt->ff.F43Rsp, &ObjectID, sizeof(ObjectID));
              leng = sizeof(ObjectID);
              addr = (uint16)&MBpkt->ff.F43Rsp + leng;
              for(j=0; j<3; j++) {
                MemCpy((uint8*)addr, &j, 1);
                addr++; leng++;
                MemCpy((uint8*)addr, &DevProfile->DevName.rec[j].len, 1);
                addr++; leng++;
                dd = DevProfile->DevName.rec[j].len;
                MemCpy((uint8*)addr, DevProfile->DevName.rec[j].str, dd);
                addr += dd;
                leng += dd;
                }
              MBpkt->MBAP.Len += leng;
              ProtectKey = 0x02;
              }
             else {
              err = MBERR_ILLEGAL_DATA_VALUE;
              ProtectKey = 0;
              }
            break;
          //---------------------------------------------------------------
          default:
            err = MBERR_ILLEGAL_FUNCTION;
            break;
          }
        if(err) {
          MBpkt->FCode |= 0x80;
          MBpkt->ff.ExeptionCode = err;
          MBpkt->MBAP.Len += 1;
          }
        NETSend(ic, s, (uint8*)MBpkt, MBpkt->MBAP.Len + sizeof(typeMBAP));
      }



      void NXXX_MainLoop() {
        uint8 ic;
        int8 s;
        int len;
        RTICommonTimerCount0 = RTISecToCount(10);
        ProtectKey = 0;
        for(;;) {
          if(NETSock_RxFlag(&ic, &s)) {
            len = NETReceive(ic, s);
            if(len >=0) nxxx_MakeResponse(ic, s, (typeMB*)Buff);
            }
          if(Sock_FindClosedSock(&ic, &s)) NETReinitSocket(ic, s);
          if(RTICommonTimerCount == 0) {
            Vars.NXXX_Var.Data.MCTmpr = LM73GetTemperature();
            NETTCPSendKeep();
            RTICommonTimerCount = RTICommonTimerCount0;
            }
          if(ProcHandledFlg) {
            ARMCOP = 0x55; ARMCOP = 0xAA;
            ProcHandledFlg &= ~1;
            }
          }
      }
```

```
-------------------------------------------------------------------
```

Nxxx_proc.c

```
#include <hidef.h>
#include <mc9s12e64.h>
#include <types.h>
#include "variable.h"

void NXXX_Proc(void);

void nxxx_ProcInit() {
  NXXX_Proc();
}


void NXXX_Proc() {
  /*Pull-up registers*/
  PERAD = Vars.NXXX_Var.Data.PurPortAD;
  PERS  =  (PERS  &  0x9F)  |  ((*((uint8*)&Vars.NXXX_Var.Data.PurPortSMPK)<<1)  &
0x60);
  PERM  =  (PERM  &  0xCF)  |  ((*((uint8*)&Vars.NXXX_Var.Data.PurPortSMPK)<<2)  &
0x30);
  PERP = (PERP & 0xFC) | (*((uint8*)&Vars.NXXX_Var.Data.PurPortSMPK) & 0x03);
  PUCR_PUPKE = (*((uint8*)&Vars.NXXX_Var.Data.PurPortSMPK+1)==0) ? 0 : 1;
  /*Direct register*/
  DDRAD = Vars.NXXX_Var.Data.DirPortAD;
  DDRS  =  (DDRS  &  0x9F)  |  ((*((uint8*)&Vars.NXXX_Var.Data.DirPortSMPK)<<1)  &
0x60);
  DDRM  =  (DDRM  &  0xCF)  |  ((*((uint8*)&Vars.NXXX_Var.Data.DirPortSMPK)<<2)  &
0x30);
  DDRP = (DDRP & 0xFC) | (*((uint8*)&Vars.NXXX_Var.Data.DirPortSMPK) & 0x03);
  DDRK = *((uint8*)&Vars.NXXX_Var.Data.DirPortSMPK+1);
  /*Data out register*/
  PTAD = Vars.NXXX_Var.Data.DataOutPortAD;
  PTS  =  (PTS  &  0x9F)  |  ((*((uint8*)&Vars.NXXX_Var.Data.DataOutPortSMPK)<<1)  &
0x60);
  PTM  =  (PTM  &  0xCF)  |  ((*((uint8*)&Vars.NXXX_Var.Data.DataOutPortSMPK)<<2)  &
0x30);
  PTP = (PTP & 0xFC) | (*((uint8*)&Vars.NXXX_Var.Data.DataOutPortSMPK) & 0x03);
  PORTK = *((uint8*)&Vars.NXXX_Var.Data.DataOutPortSMPK+1);
  /*Data in register*/
  Vars.NXXX_Var.Data.DataInPortAD = PTIAD;
  *((uint8*)&Vars.NXXX_Var.Data.DataInPortSMPK+1) = PORTK;
  *((uint8*)&Vars.NXXX_Var.Data.DataInPortSMPK)                              =
((PTIS>>1)&0x30)|((PTIM>>2)&0x0C)|(PTIP&0x03);
}


void NXXX_IntRTIHandlerProc() {
  ProcHandledFlg = 1;
}


void NXXX_IntIRQHandlerProc() {
}
```

Таблица регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в док.	Номер док.	Подп.	Дата
	изме-ненных	заме-ненных	новых	аннули-рован-ных				