Приложение

Пошагово посмотрим, как работает микроконтроллер. Зададим начальные параметры:

```
// Глобальные переменные
byte frameFromArbiter = 0;
bool abonentStateReceived = false;
bool frameCameFromArbiter = true;
bool AddressCameFromArbiter = true;
bool frameToAbonentWasSent = true;
bool requestToAbonentWasSent = true;
bool stateOfAbonentWasSentToArbiter = true;
bool DataMessageRefered = true;
int abonentState = 0;
// Ждём поступления от Арбитра адреса микроконтроллера
void WaitAddressFromArbiter()
{
DDRA = 0x00; // Указываем микроконтроллеру настроить все выводы на приём информации
bool attentionCame = true;
// Посылаем параллельно байт Абоненту
void SendFrameToAbonent(byte sendFrame)
{
static bool C1Came = true;
if(C1) // Если пришёл сигнал С 1,
{
C1Came = true; // запоминаем что сигнал С 1 приходил
}
// Получаем от Абонента его состояние
void ReceiveFromAbonentAbonentState()
```

{

DDRB = 0x00; // Указываем микроконтроллеру настроить все выводы порта B на приём информации

static bool C1Came = true;

// Посылаем Арбитру состояние Абонента

void SendToArbiterAbonentState(byte frame)

{

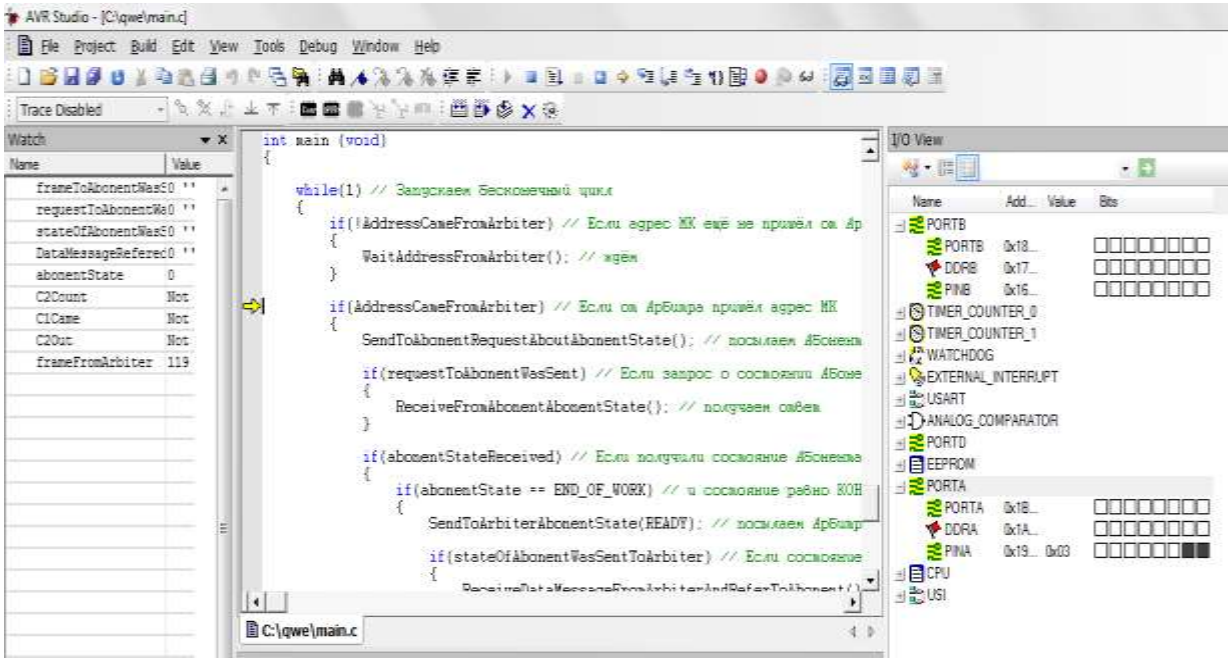
DDRA = 0x04;

static int C2Count = 1;

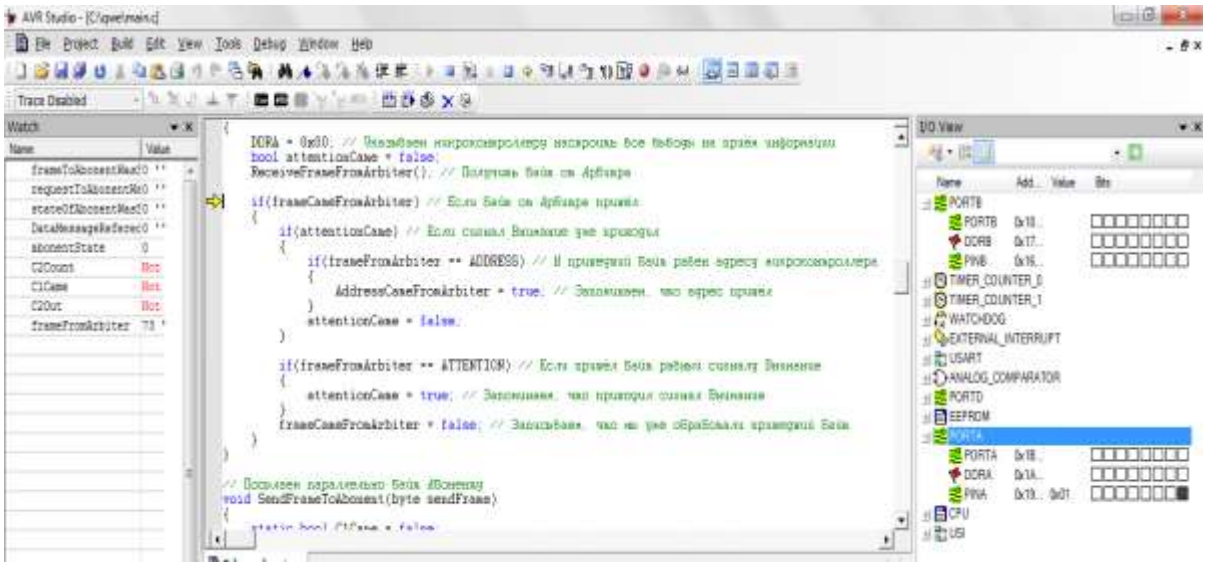
static bool C1Came = true;

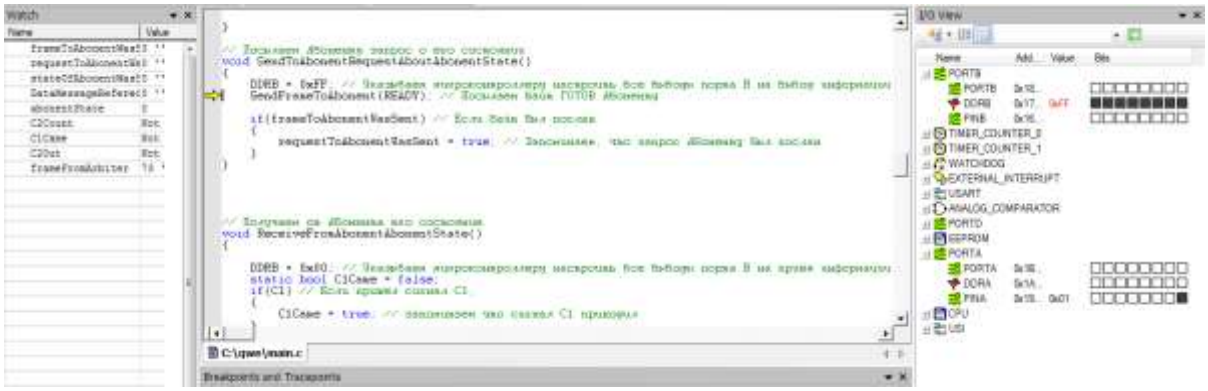
static bool C2Out = true;

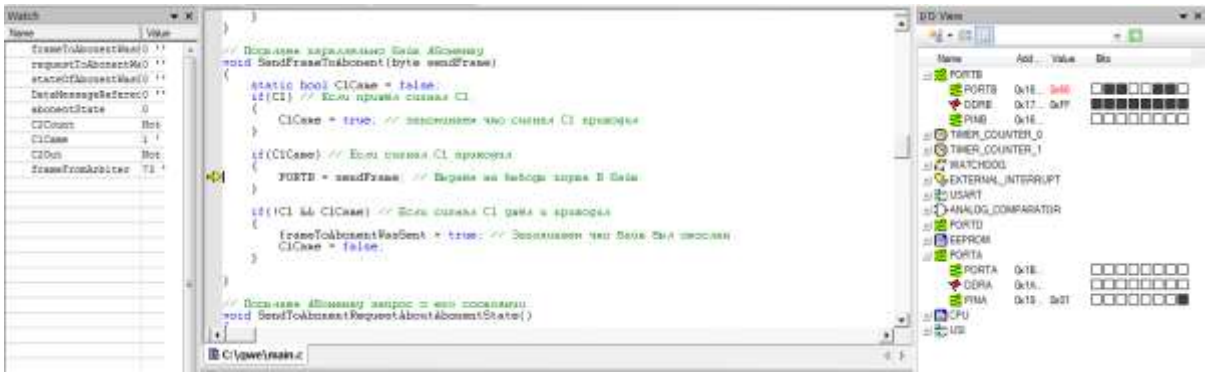
Арбитр вырабатывает общий для всех абонентов сигнал ВНИМАНИЕ:

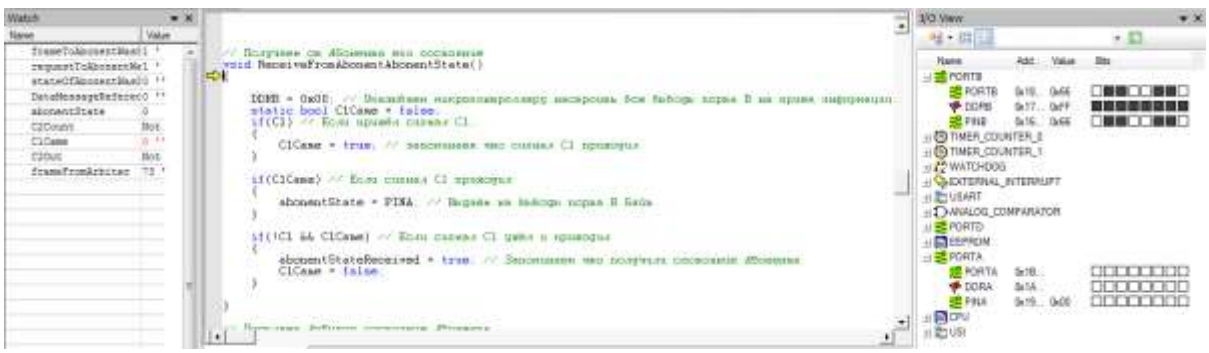


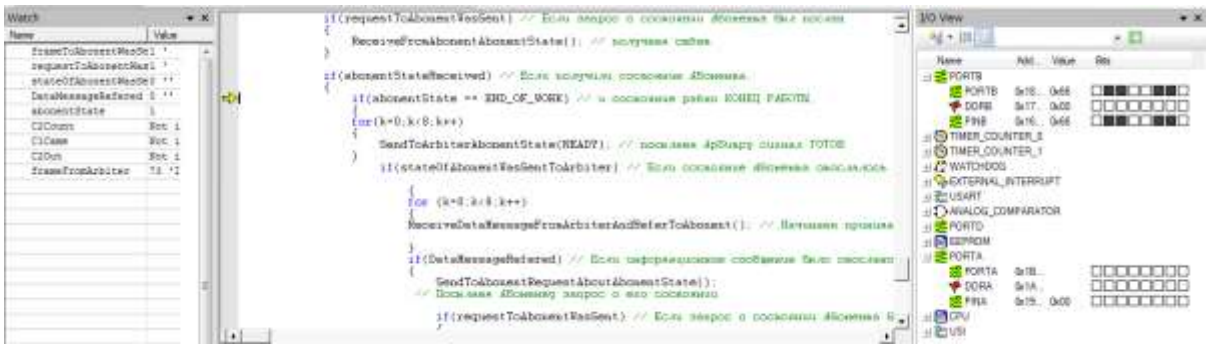
Арбитр вырабатывает общий для всех абонентов сигнал ВНИМАНИЕ и затем – АДРЕС нужного абонента.

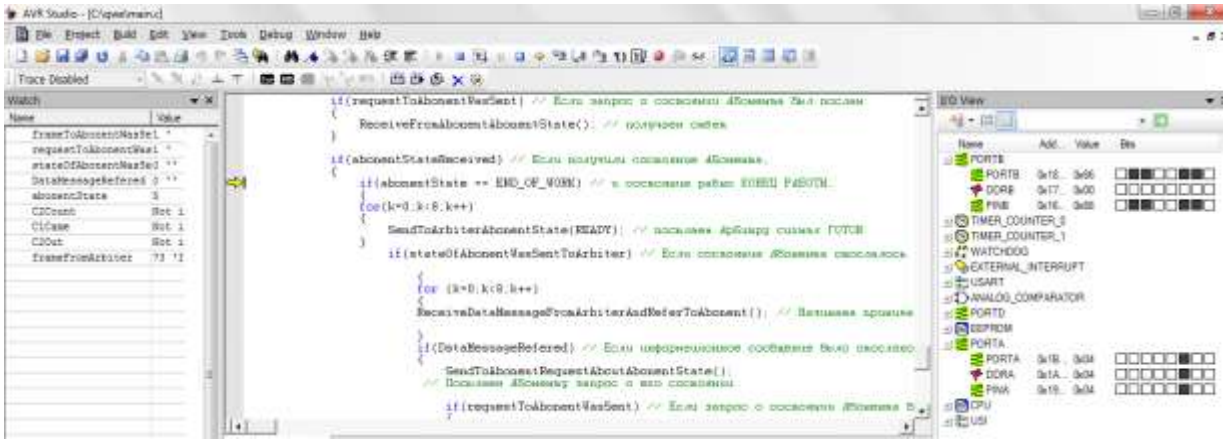

AVR Studio - [C:\qwe\main.c]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Watch
Name Value
I/O View
Name Add... Value Bits
PORTB
PORTB 0x18..
DDRB 0x17..
PINB 0x16..
TIMER_COUNTER_0
TIMER_COUNTER_1
WATCHDOG
EXTERNAL_INTERRUPT
USART
ANALOG_COMPARATOR
PORTD
EEPROM
PORTA
PORTA 0x1B..
DDRA 0x1A..
PINA 0x19.. 0x01
CPU
USI
DDRA = 0x00; // Указываем микроконтроллеру настроить все выводы на прием информации
bool attentionCame = false;
ReceiveFrameFromArbiter(); // Получаем байт от Арбитра
if(frameCameFromArbiter) // Если байт от Арбитра принят
if(attentionCame) // Если сигнал Внимание уже приходил
if(frameFromArbiter == ADDRESS) // И принятый байт равен адресу микроконтроллера
AddressCameFromArbiter = true; // Запоминаем, что адрес принят
attentionCame = false;
if(frameFromArbiter == ATTENTION) // Если принят байт равный сигналу Внимание
attentionCame = true; // Запоминаем, что приходил сигнал Внимание
frameCameFromArbiter = false; // Запоминаем, что мы уже обработали принятый байт
// Посылаем параллельно байт абоненту
void SendFrameToAbonent(byte sendFrame)

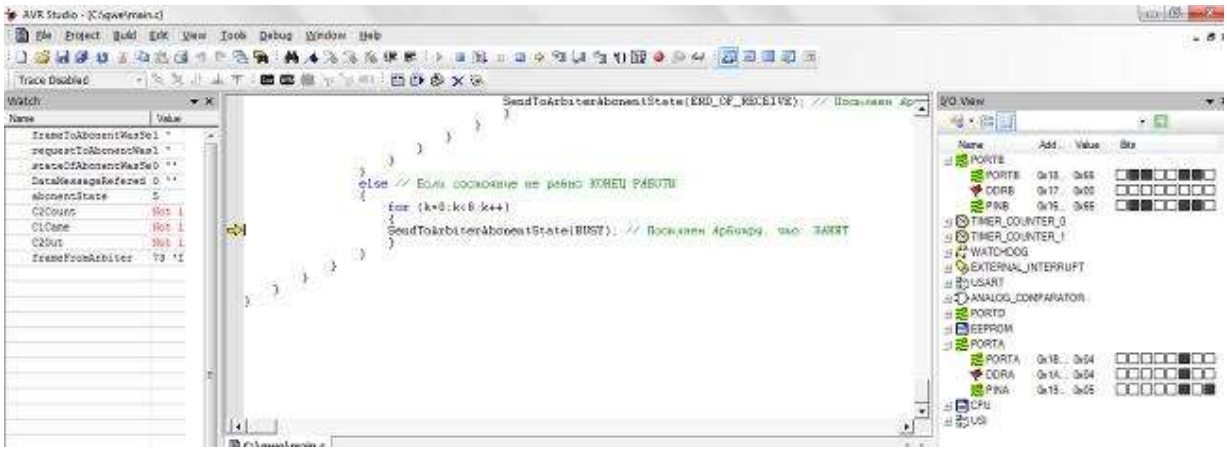

Watch
Name Value
I/O View
Name Add... Value Bits
PORTB
PORTB 0x18..
DDRB 0x17.. 0xFF
PINB 0x16..
TIMER_COUNTER_0
TIMER_COUNTER_1
WATCHDOG
EXTERNAL_INTERRUPT
USART
ANALOG_COMPARATOR
PORTD
EEPROM
PORTA
PORTA 0x1B..
DDRA 0x1A..
PINA 0x19.. 0x01
CPU
USI
// Посылаем абоненту запрос о его состоянии
void SendToAbonentRequestAboutAbonentState()
DDRB = 0xFF; // Указываем микроконтроллеру настроить все выводы порта B на вывод информации
SendFrameToAbonent(READY); // Посылаем байт ГОТОВ абоненту
if(frameToAbonentWasSent) // Если байт был послан
requestToAbonentWasSent = true; // Запоминаем, что запрос абоненту был послан
// Получаем от абонента его состояние
void ReceiveFromAbonentAbonentState()
DDRB = 0x00; // Указываем микроконтроллеру настроить все выводы порта B на прием информации
static bool C1Came = false;
if(C1) // Если пришел сигнал C1
C1Came = true; // запоминаем что сигнал C1 пришел
C:\qwe\main.c
Breakpoints and Tracepoints


Watch
Name Value
I/O View
Name Add... Value Bits
PORTB
PORTB 0x18.. 0x00
DDRB 0x17.. 0xFF
PINB 0x16..
TIMER_COUNTER_0
TIMER_COUNTER_1
WATCHDOG
EXTERNAL_INTERRUPT
USART
ANALOG_COMPARATOR
PORTD
EEPROM
PORTA
PORTA 0x1B..
DDRA 0x1A..
PINA 0x19.. 0x01
CPU
USI
// Посылаем параллельно байт абоненту
void SendToAbonent(byte sendFrame)
static bool C1Came = false;
if(C1) // Если пришел сигнал C1
C1Came = true; // запоминаем что сигнал C1 пришел
if(C1Came) // Если сигнал C1 пришел
PORTB = sendFrame; // Выдаем на выводы порта B байт
if(!C1 && C1Came) // Если сигнал C1 ушел и пришел
frameToAbonentWasSent = true; // Запоминаем что байт был послан
C1Came = false;
// Посылаем абоненту запрос о его состоянии
void SendToAbonentRequestAboutAbonentState()
C:\qwe\main.c


Watch
Name Value
I/O View
Name Add... Value Bits
PORTB
PORTB 0x18.. 0x66
DDRB 0x17.. 0xFF
PINB 0x16.. 0x66
TIMER_COUNTER_0
TIMER_COUNTER_1
WATCHDOG
EXTERNAL_INTERRUPT
USART
ANALOG_COMPARATOR
PORTD
EEPROM
PORTA
PORTA 0x1B..
DDRA 0x1A..
PINA 0x19.. 0x00
CPU
USI
// Получаем от абонента его состояние
void ReceiveFromAbonentAbonentState()
DDRB = 0x00; // Указываем микроконтроллеру настроить все выводы порта B на прием информации
static bool C1Came = false;
if(C1) // Если пришел сигнал C1
C1Came = true; // запоминаем что сигнал C1 пришел
if(C1Came) // Если сигнал C1 пришел
abonentState = PINA; // Выдаем на выводы порта B байт
if(!C1 && C1Came) // Если сигнал C1 ушел и пришел
abonentStateReceived = true; // Запоминаем что получили состояние Абонента
C1Came = false;

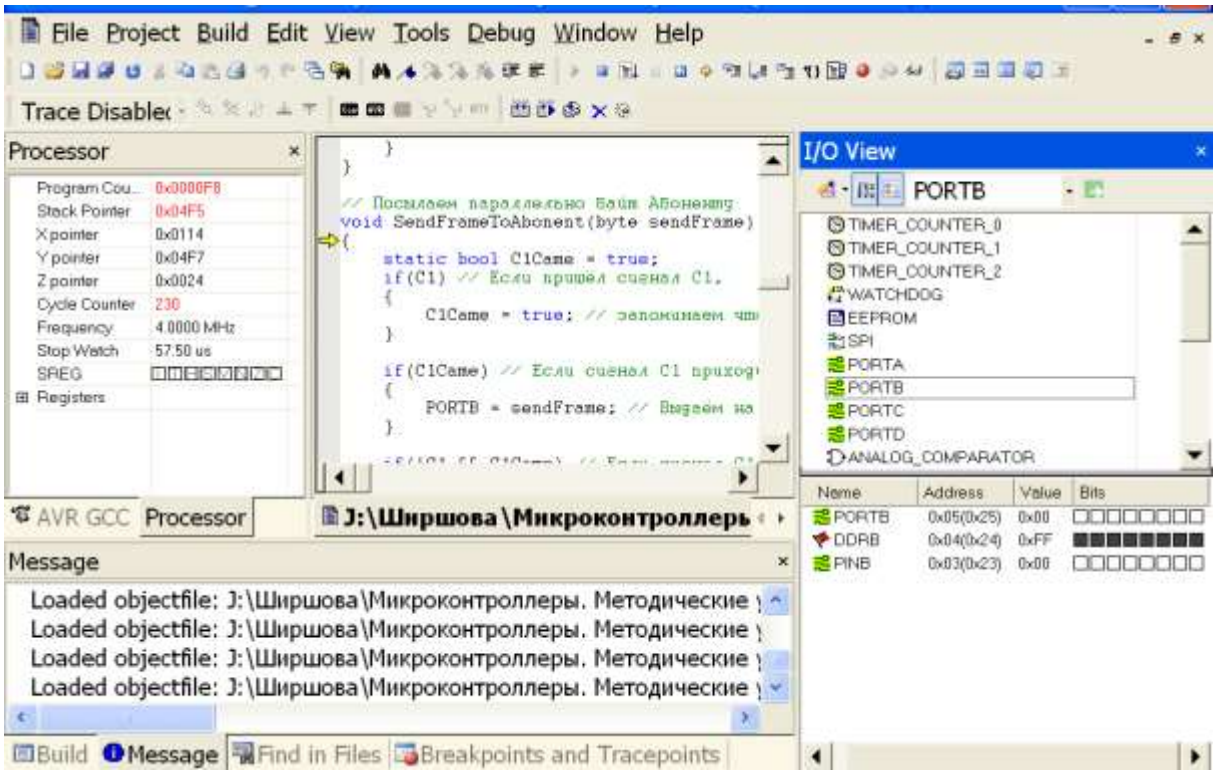




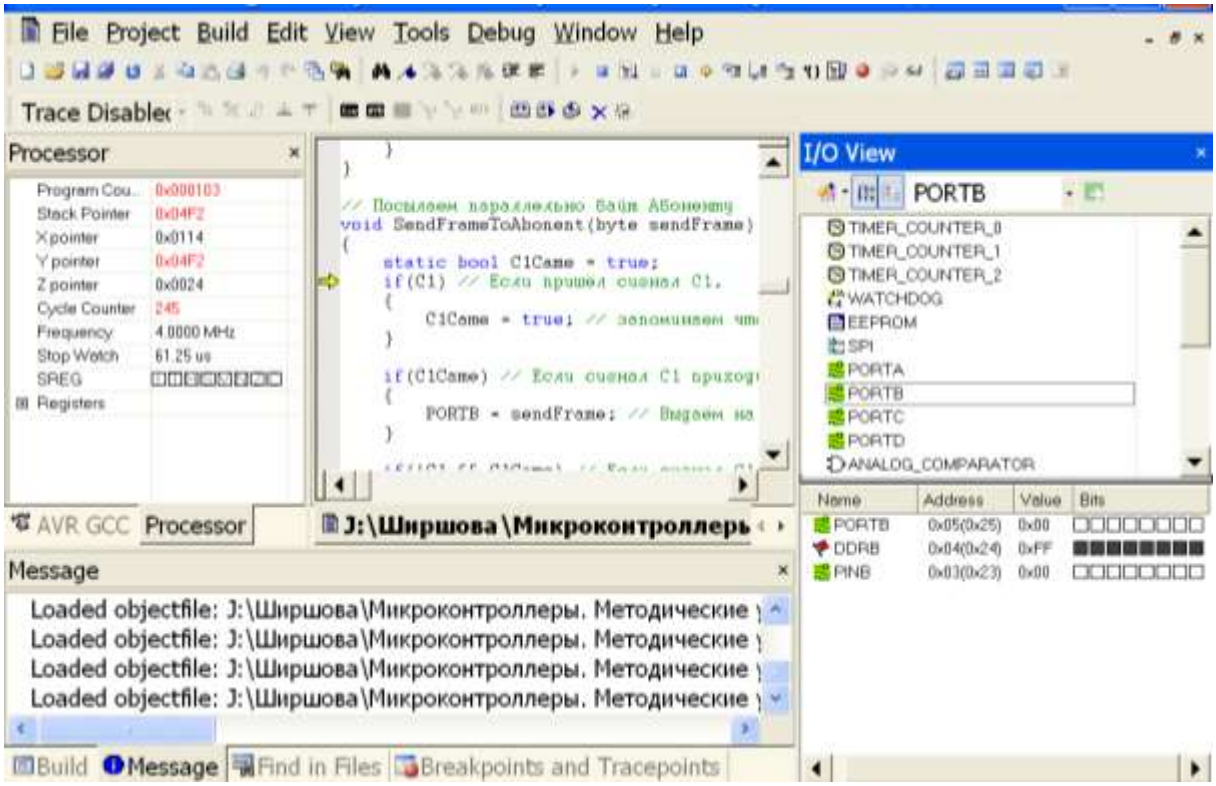


// Посылаем параллельно байт Абоненту

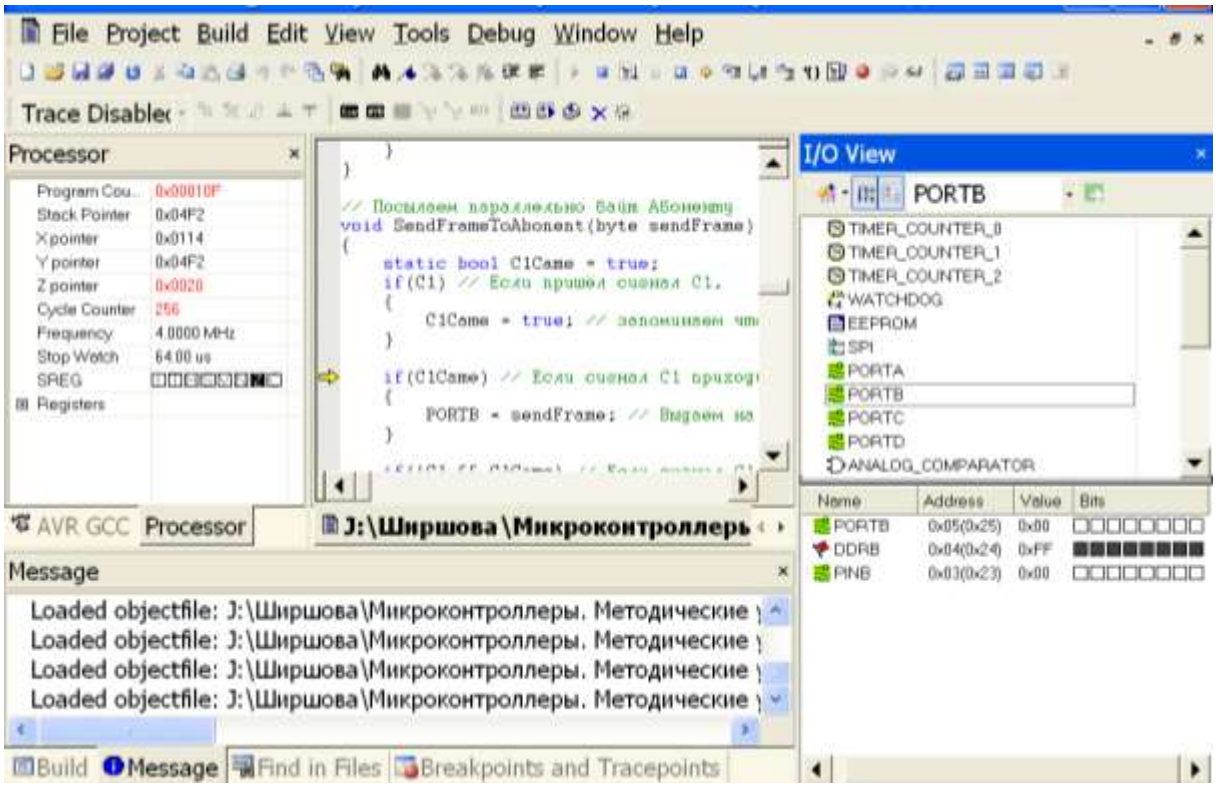
void SendFrameToAbonent(byte sendFrame)



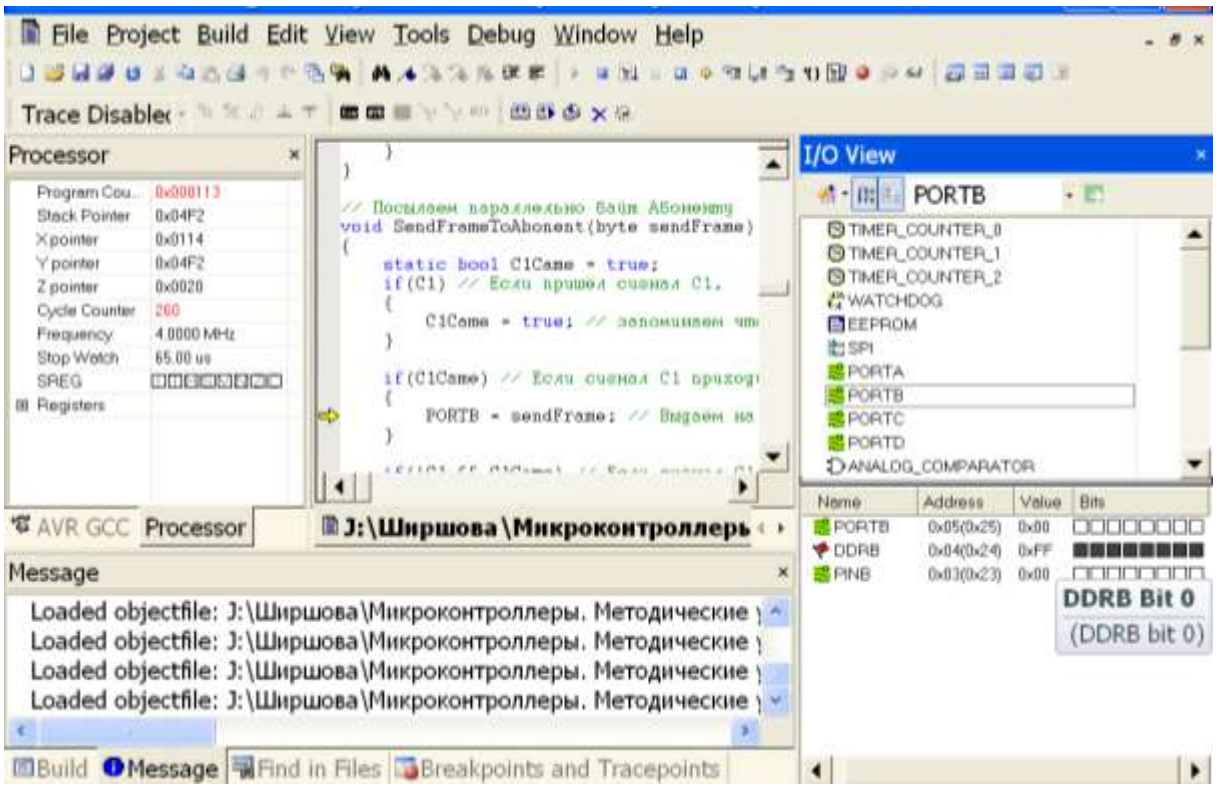
if(C1) // Если пришёл сигнал С 1,



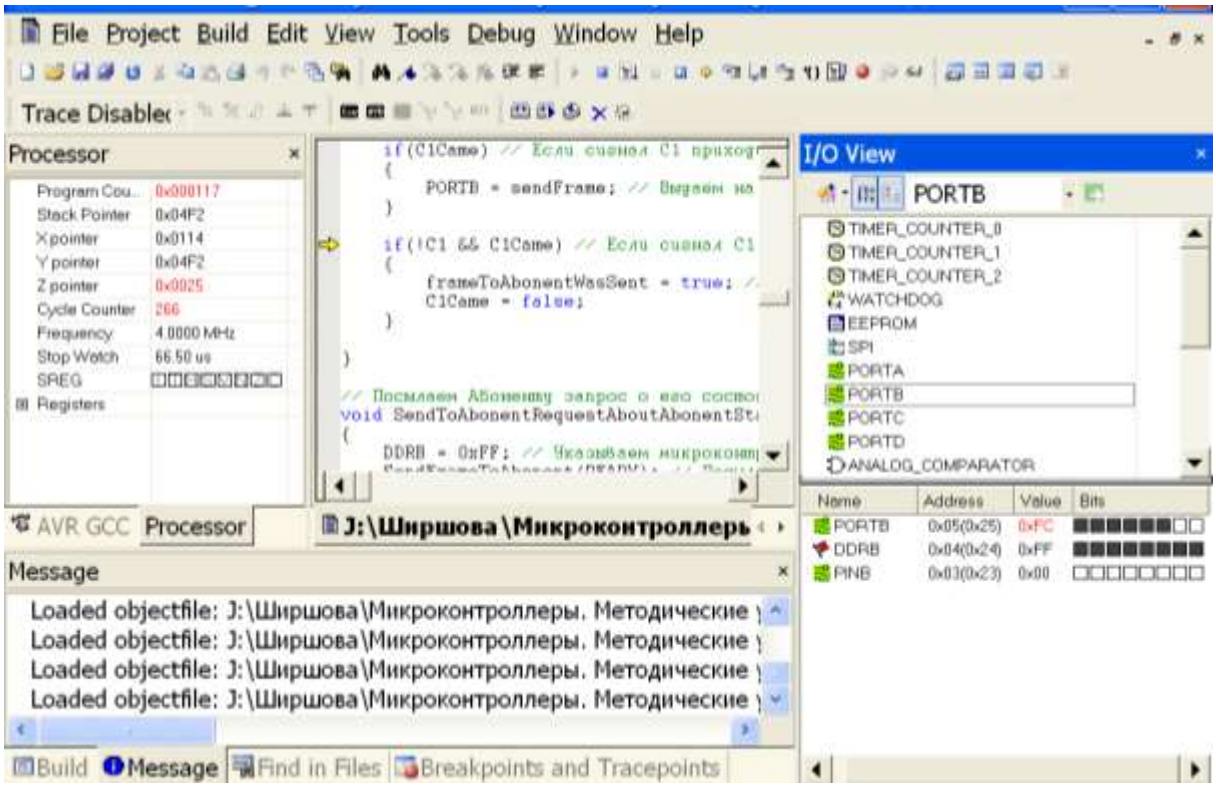
if(C1Came) // Если сигнал С 1 приходил



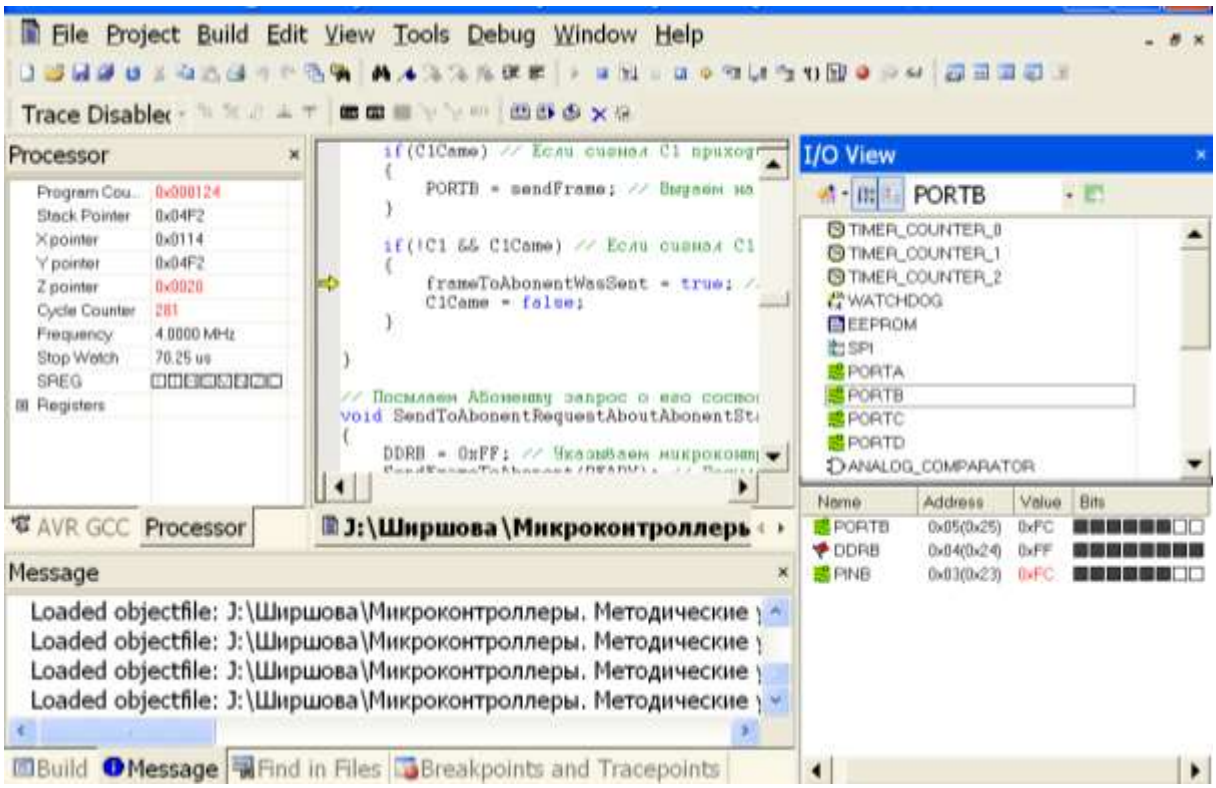
PORTB = sendFrame; // Выдаём на выводы порта B байт



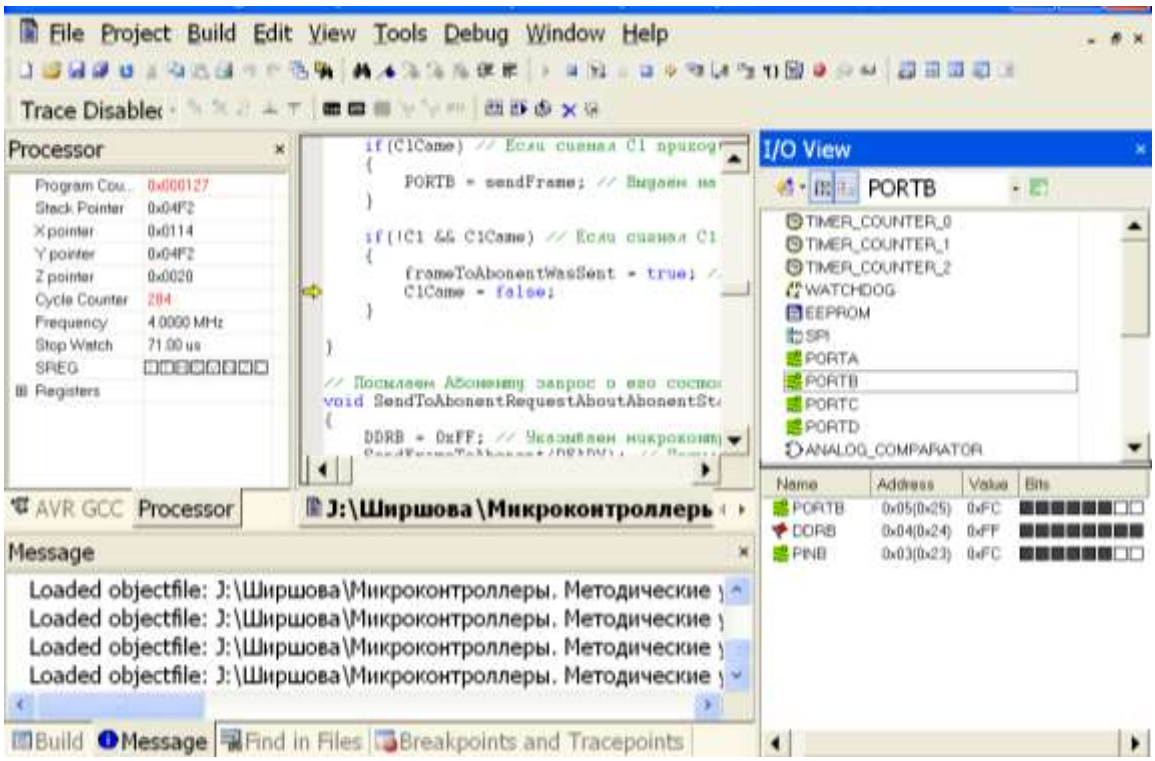
if(!C1 && C1Came) // Если сигнал С 1 ушёл и приходил



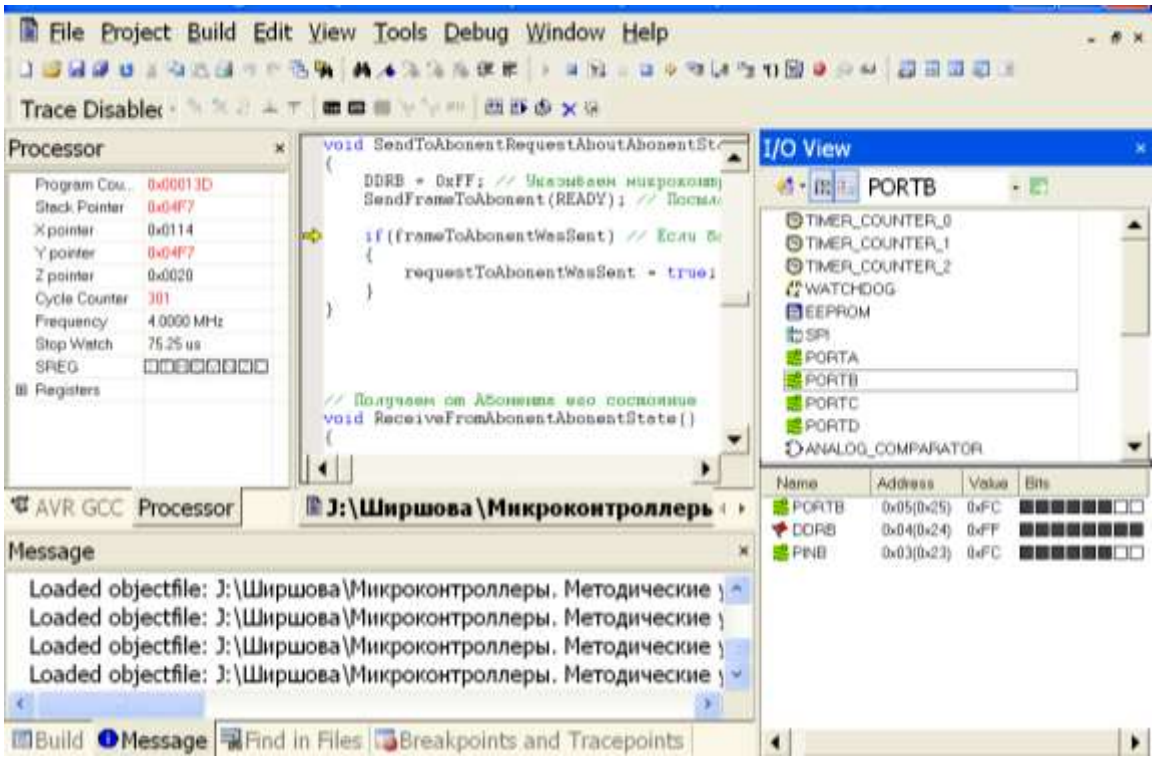
frameToAbonentWasSent = true; // Запоминаем что байт был отослан



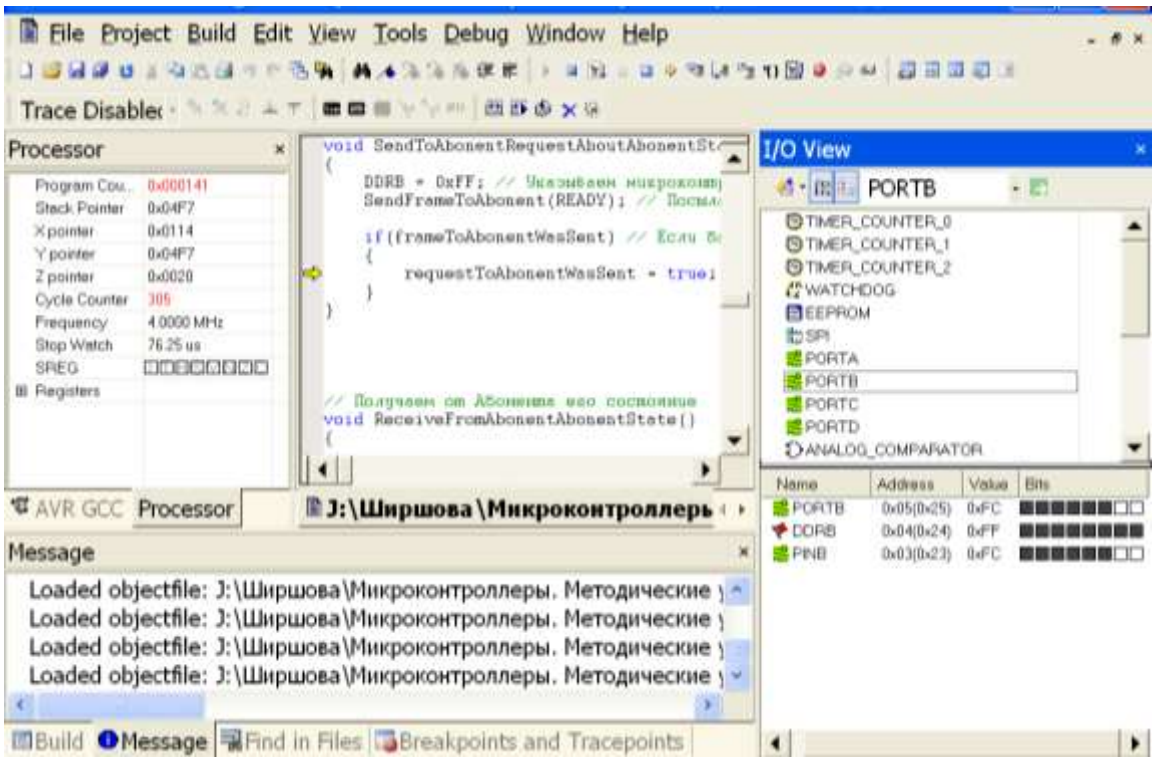
C1Came = false;



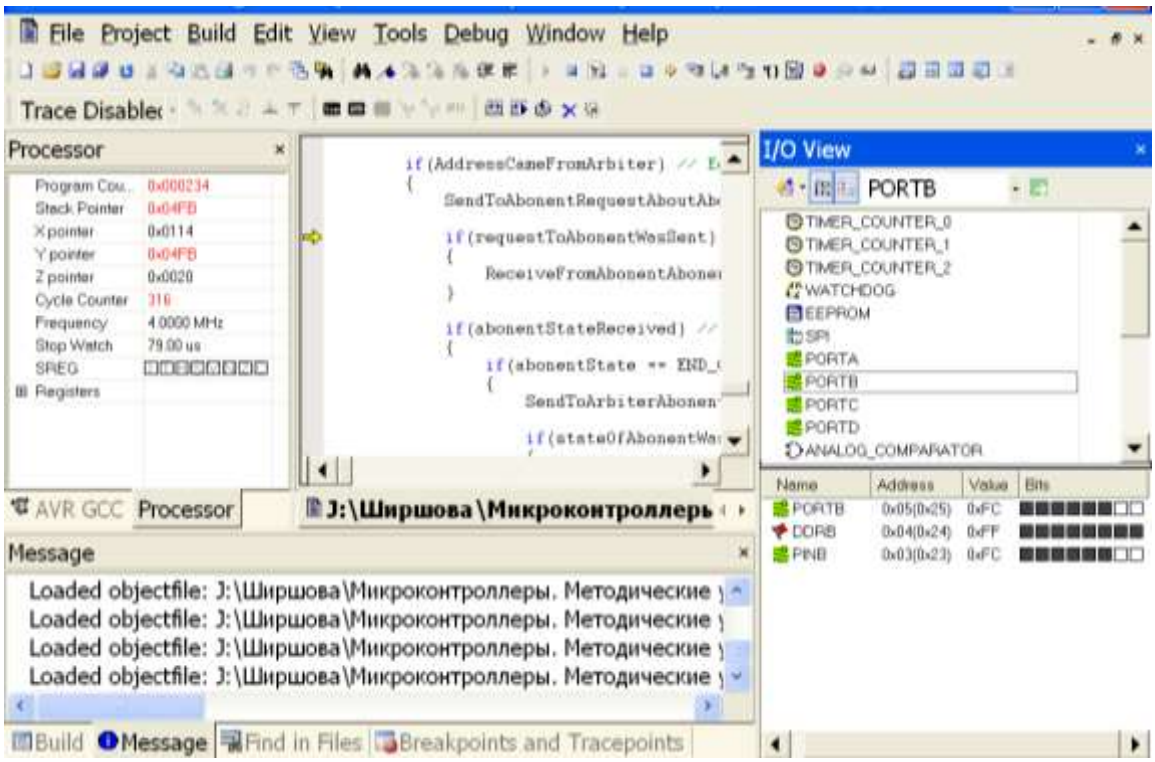
if(frameToAbonentWasSent) // Если байт был послан



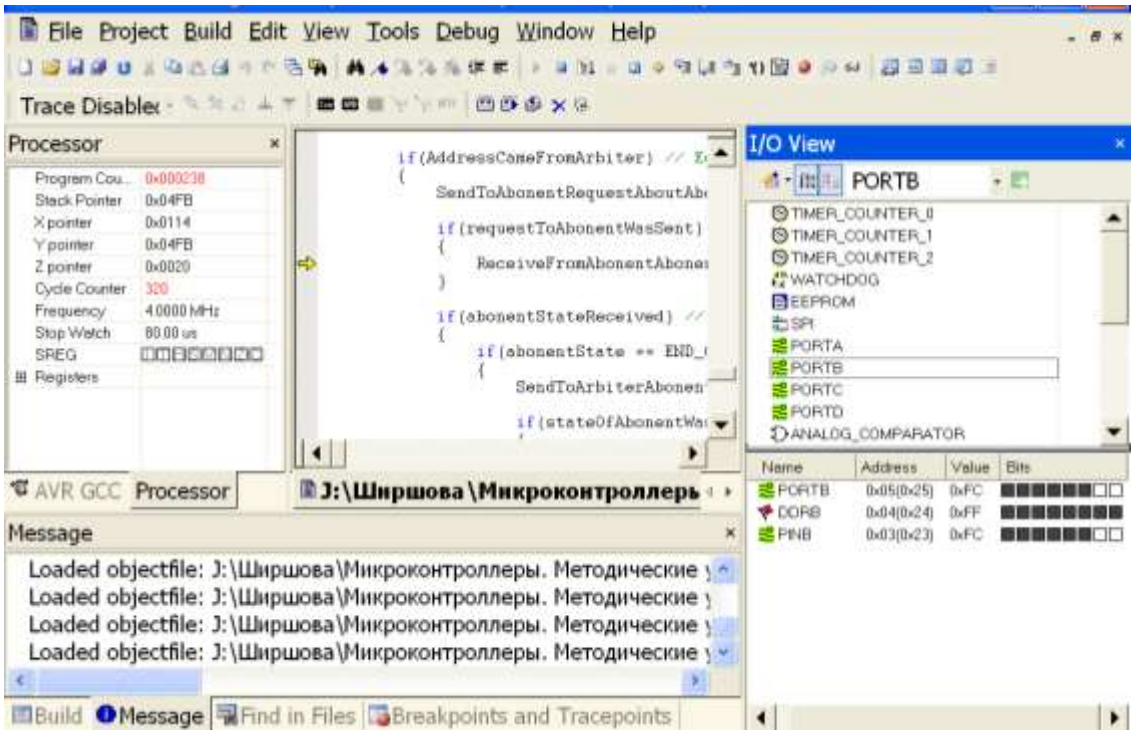
requestToAbonentWasSent = true; // Запоминаем, что запрос Абоненту был послан



if(requestToAbonentWasSent) // Если запрос о состоянии Абонента был послан

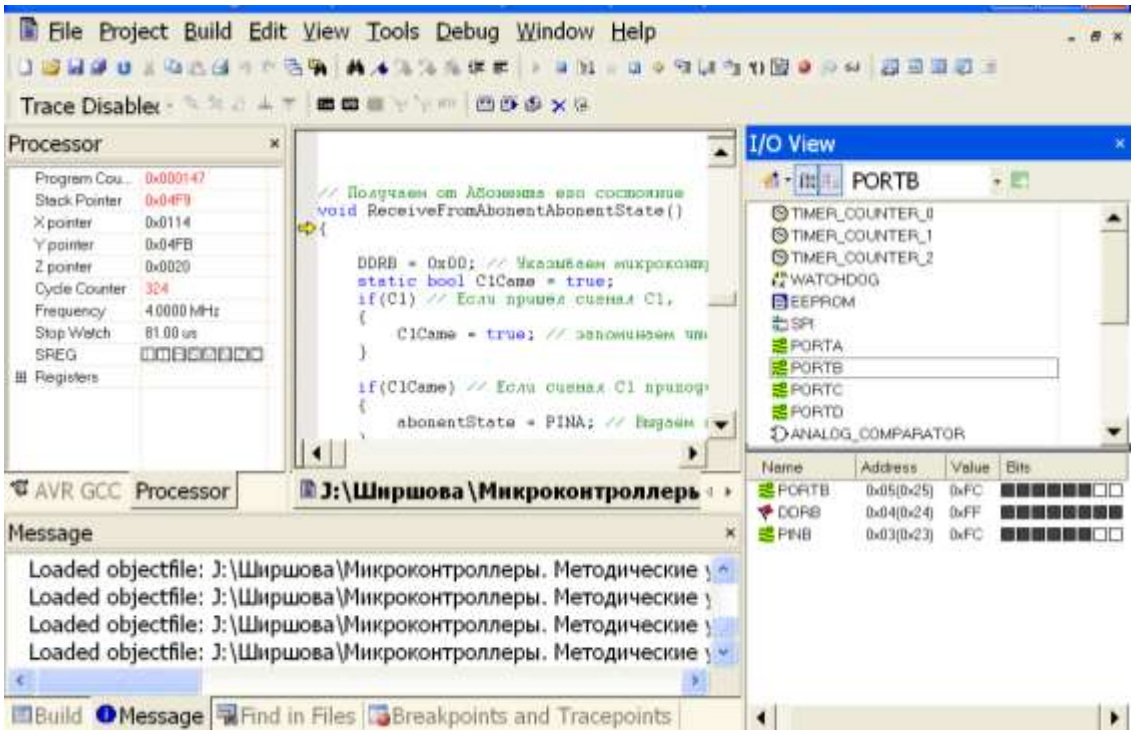


ReceiveFromAbonentAbonentState(); // получаем ответ

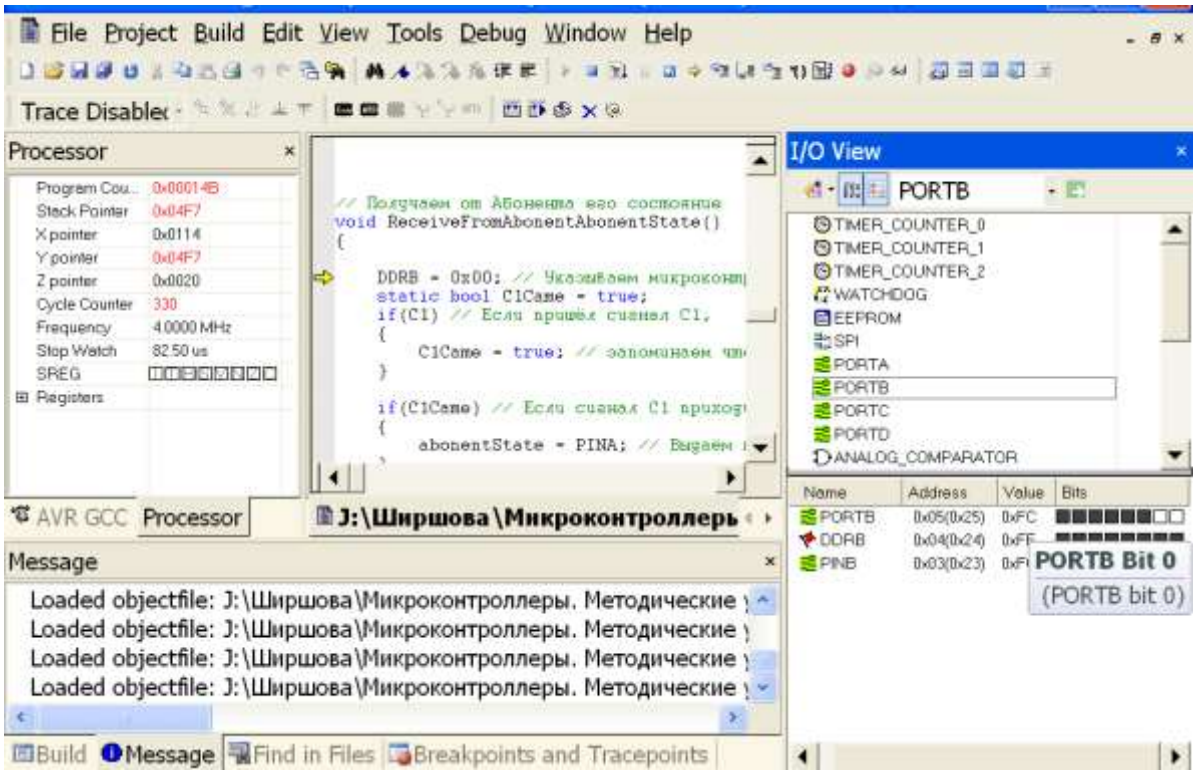


// Получаем от Абонента его состояние

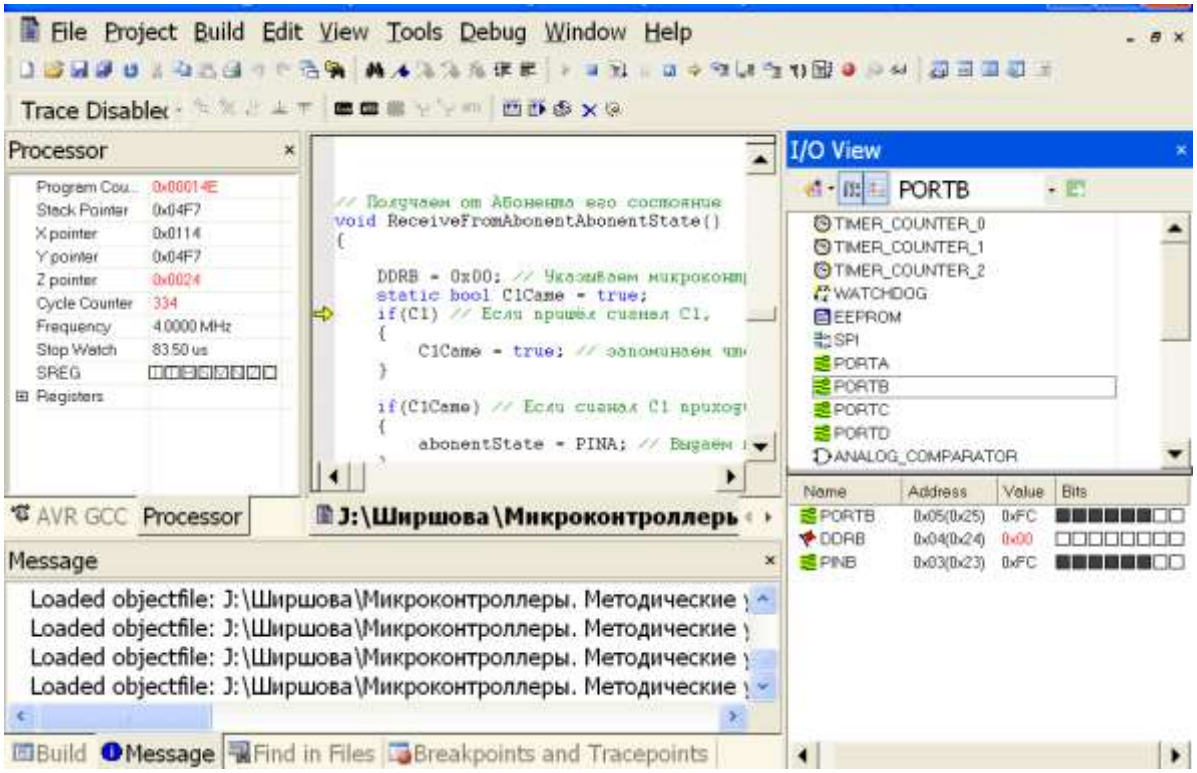
void ReceiveFromAbonentAbonentState()



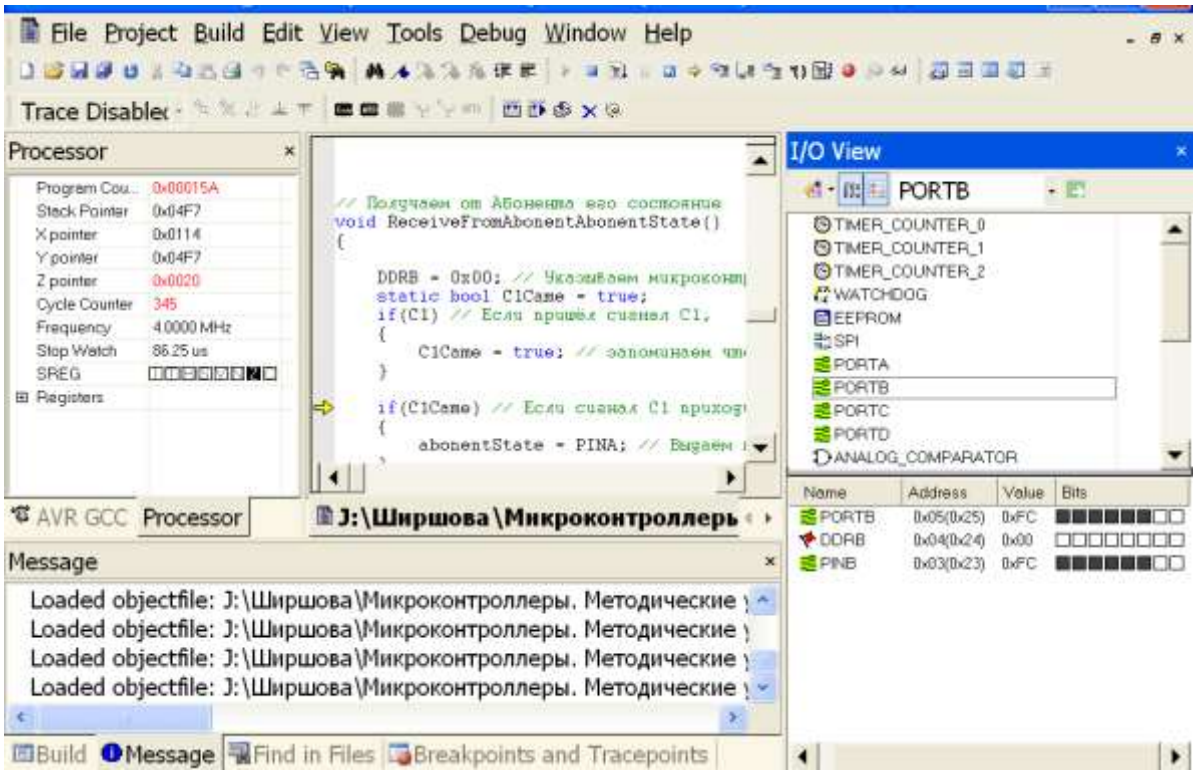
DDRB = 0x00; // Указываем микроконтроллеру настроить все выводы порта B на приём информации



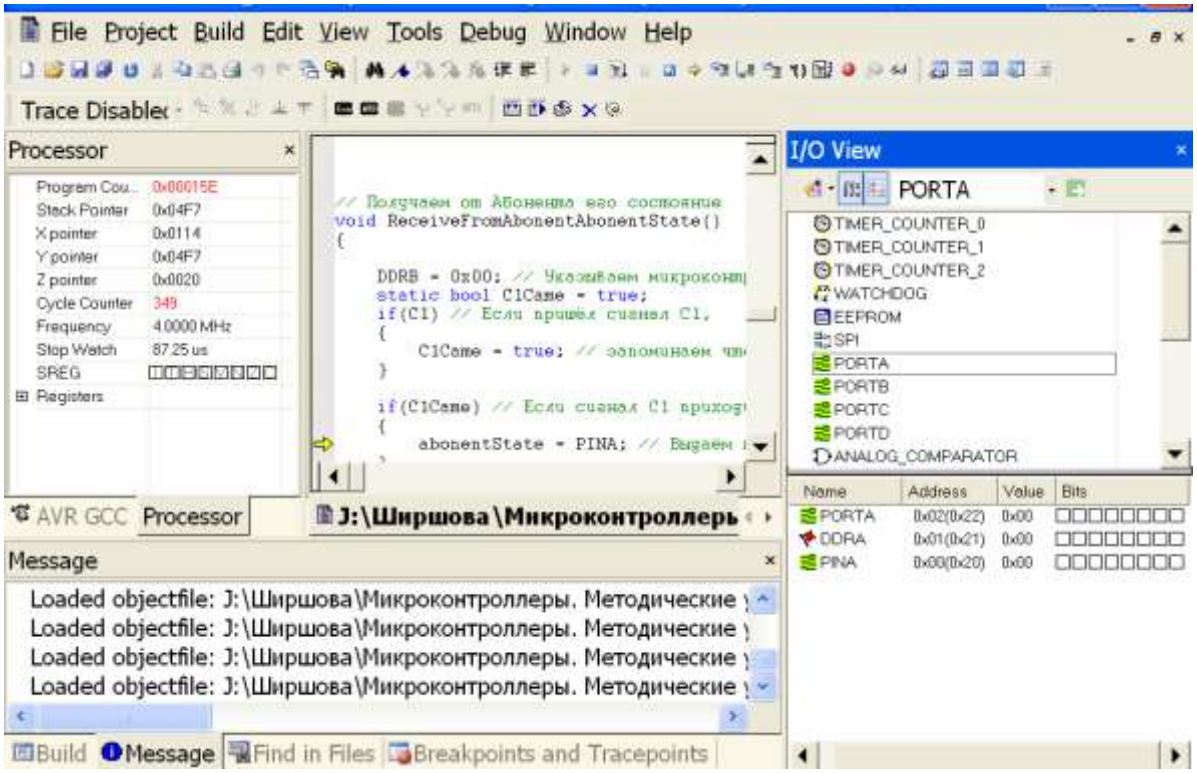
if(C1) // Если пришёл сигнал С 1,



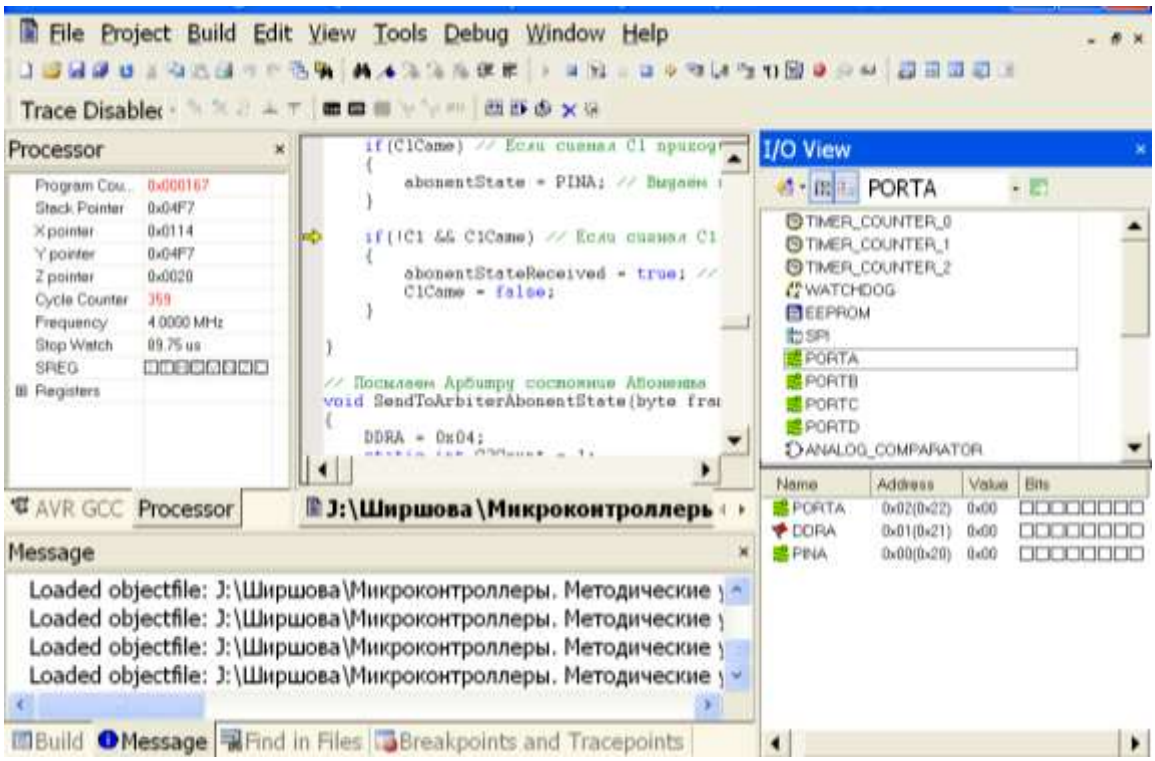
if(C1Came) // Если сигнал С 1 приходил



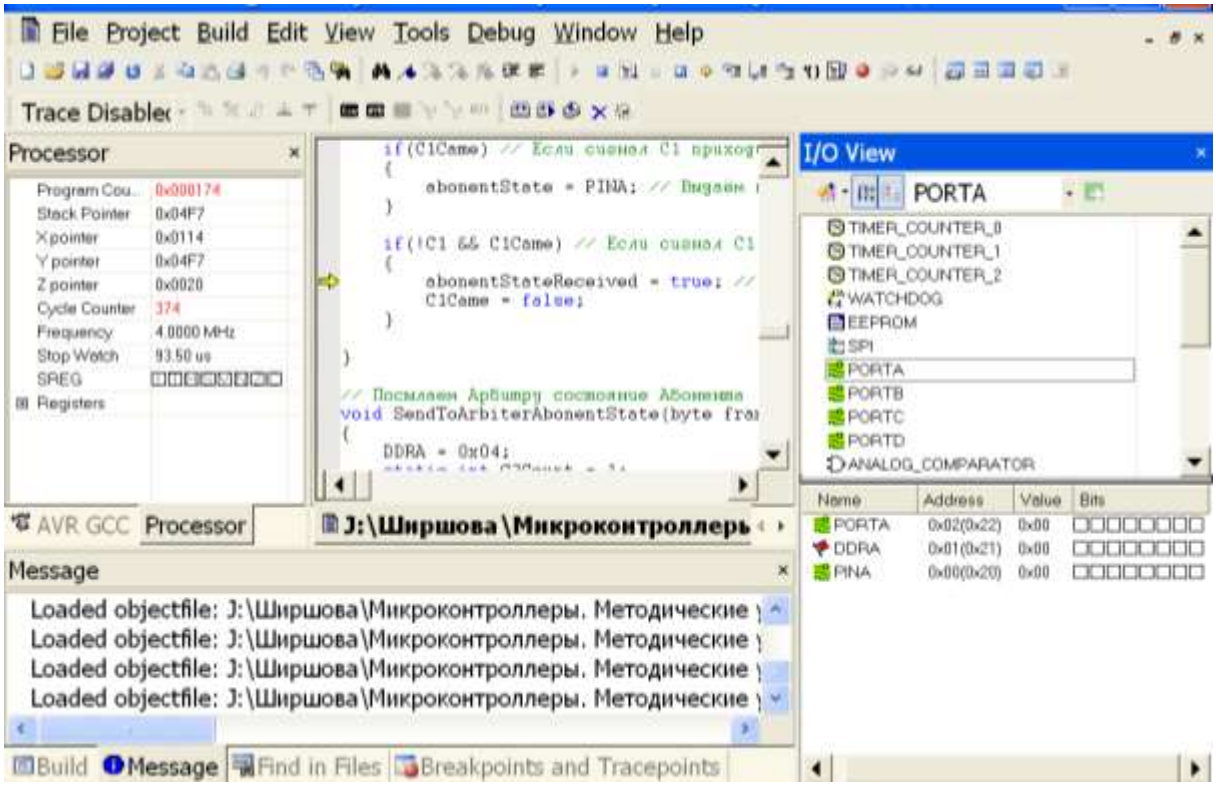
abonentState = PINA; // Выдаём на выводы порта B байт



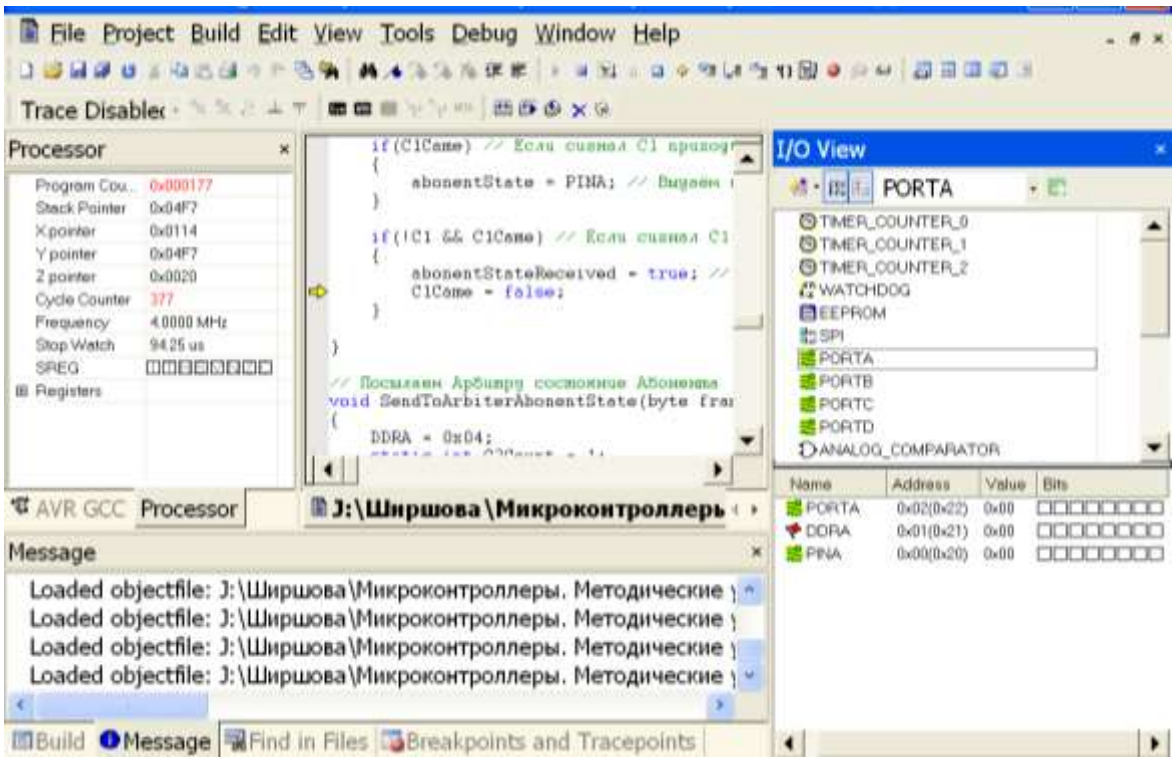
if(!C1 && C1Came) // Если сигнал C 1 ушёл и приходил



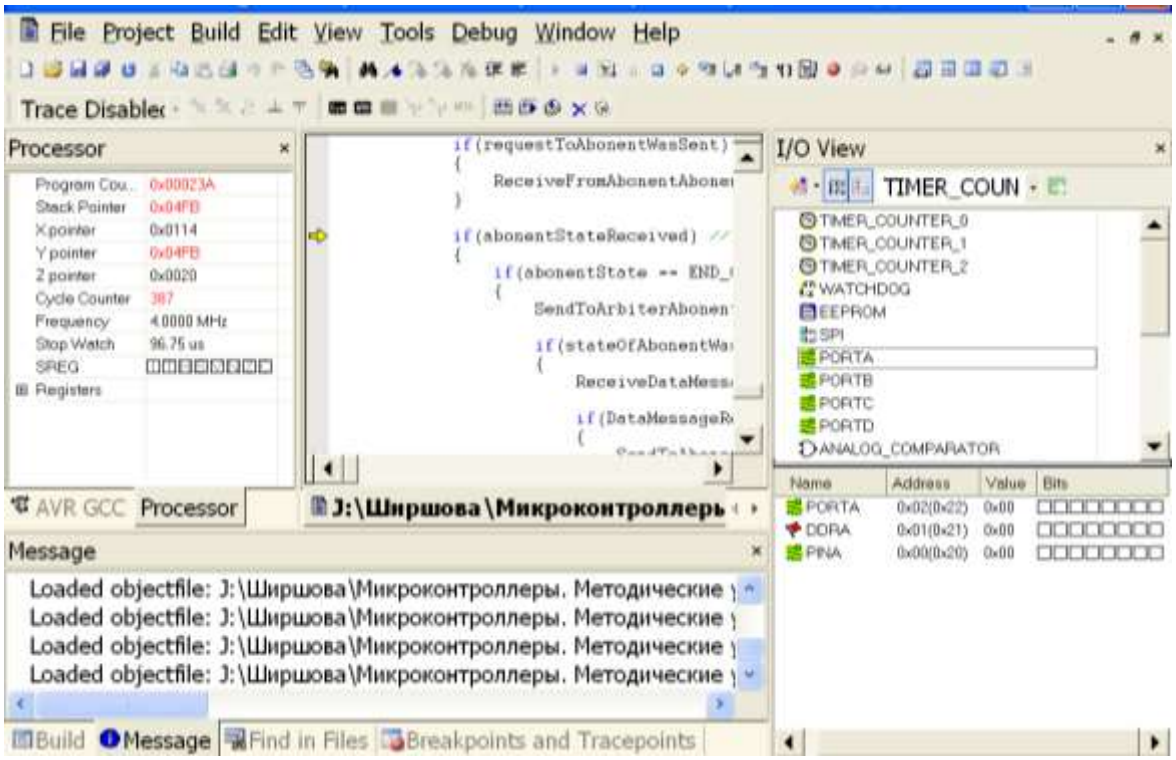
abonentStateReceived = true; // Запоминаем что получили состояние Абонента



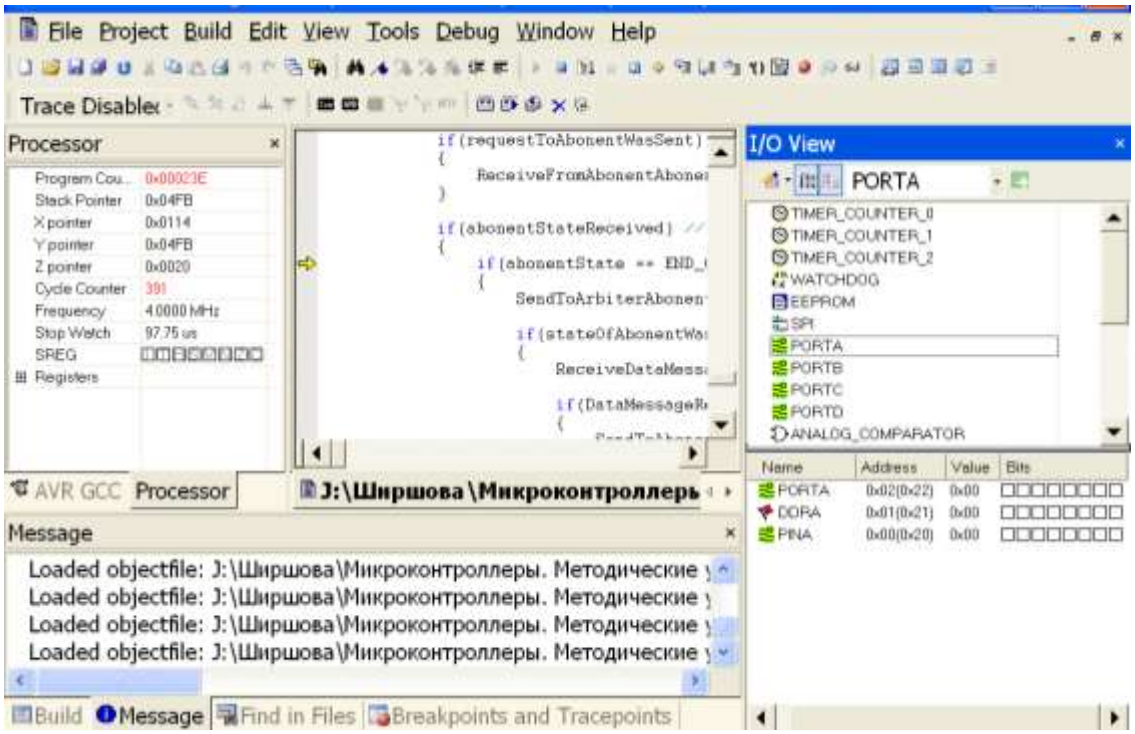
C1Came = false;



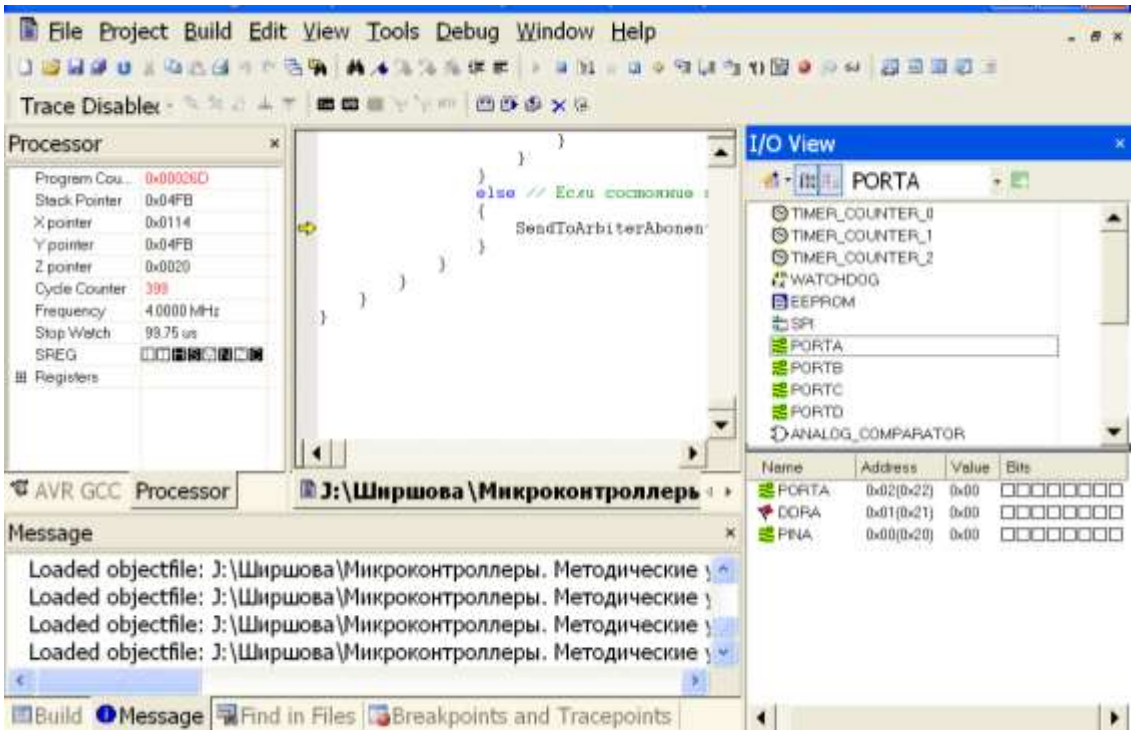
if(abonentStateReceived) // Если получили состояние Абонента,



if(abonentState == END_OF_WORK) // и состояние равно КОНЕЦ РАБОТЫ,

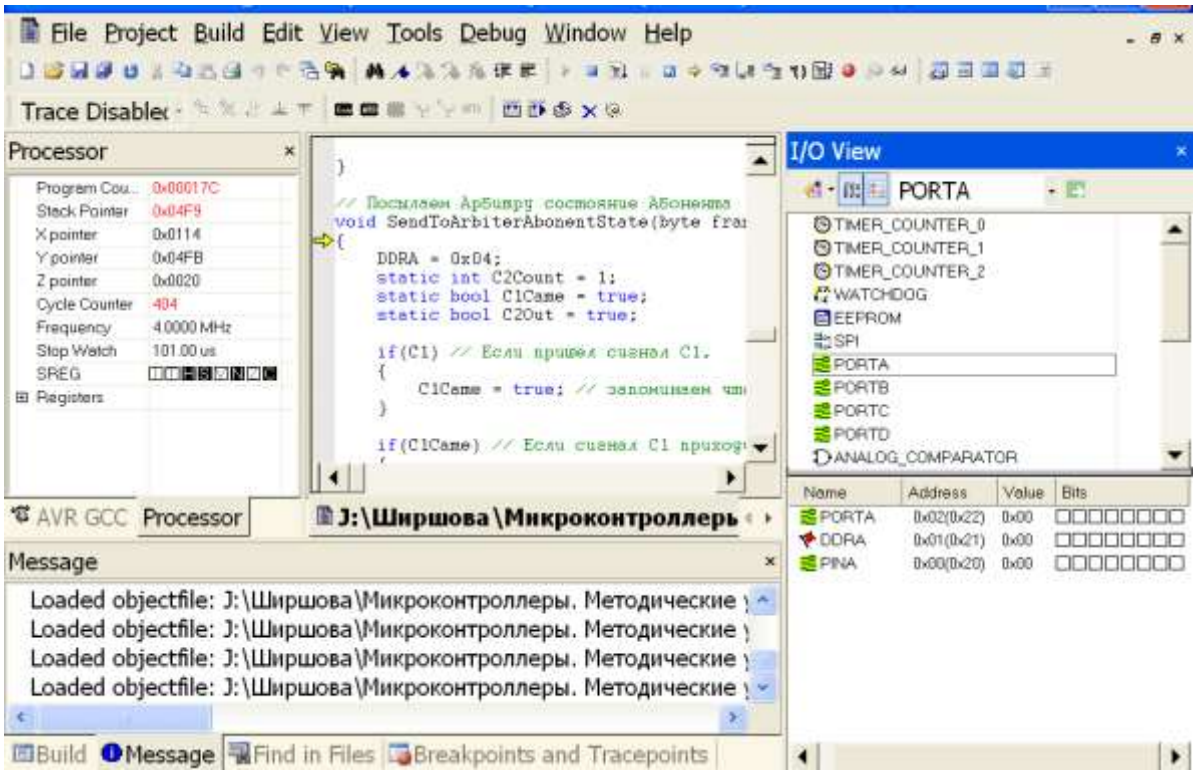


SendToArbiterAbonentState(BUSY); // Посылаем Арбитру, что Абонент ЗАНЯТ

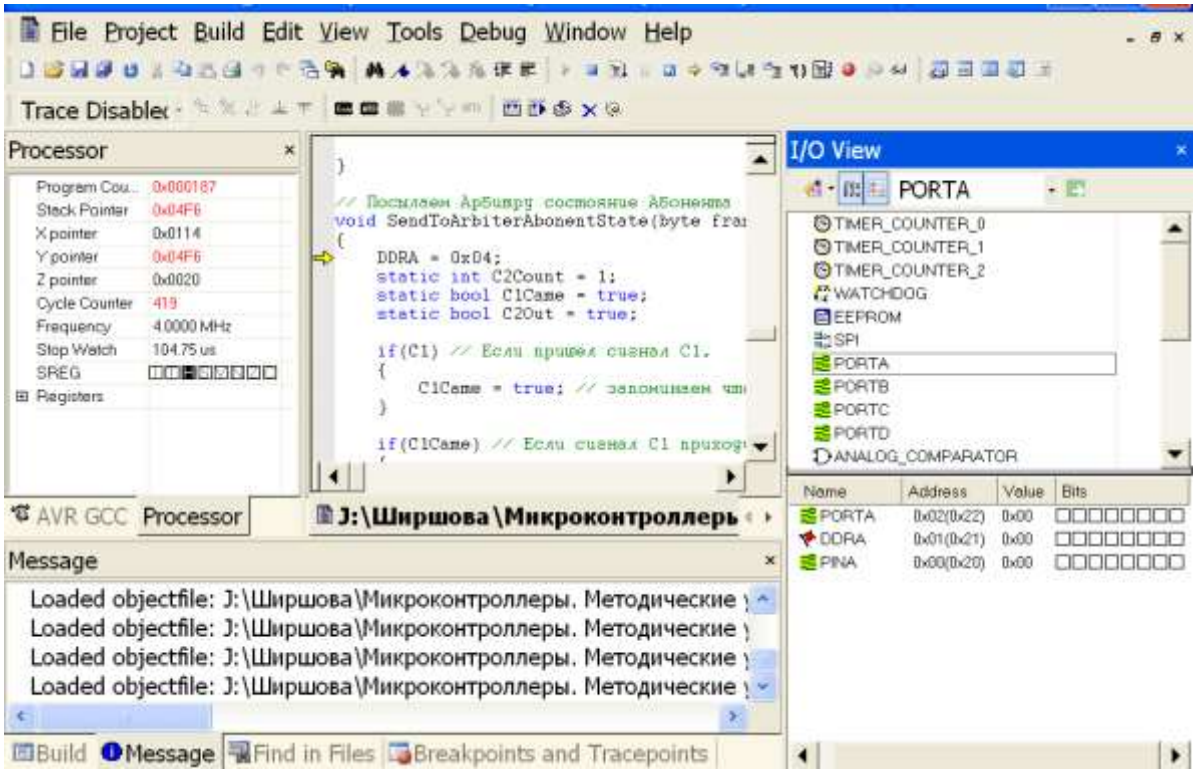


// Посылаем Арбитру состояние Абонента

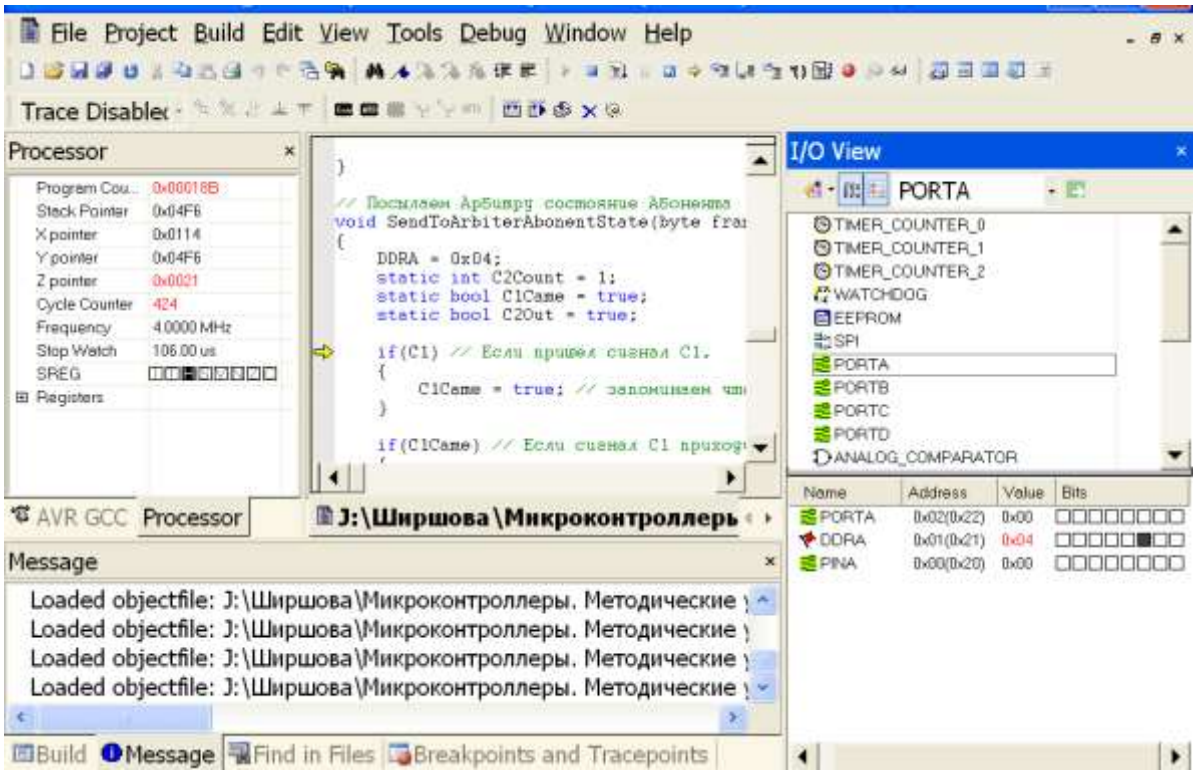
void SendToArbiterAbonentState(byte frame)



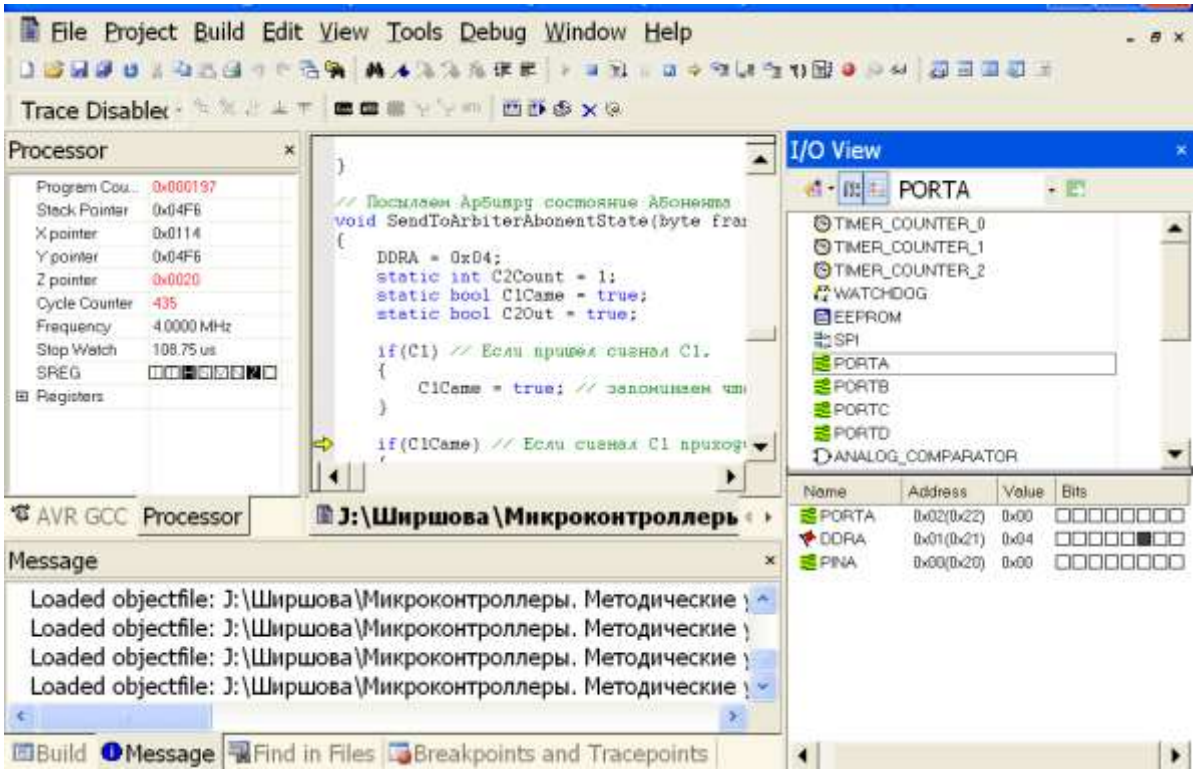
DDRA = 0x04;



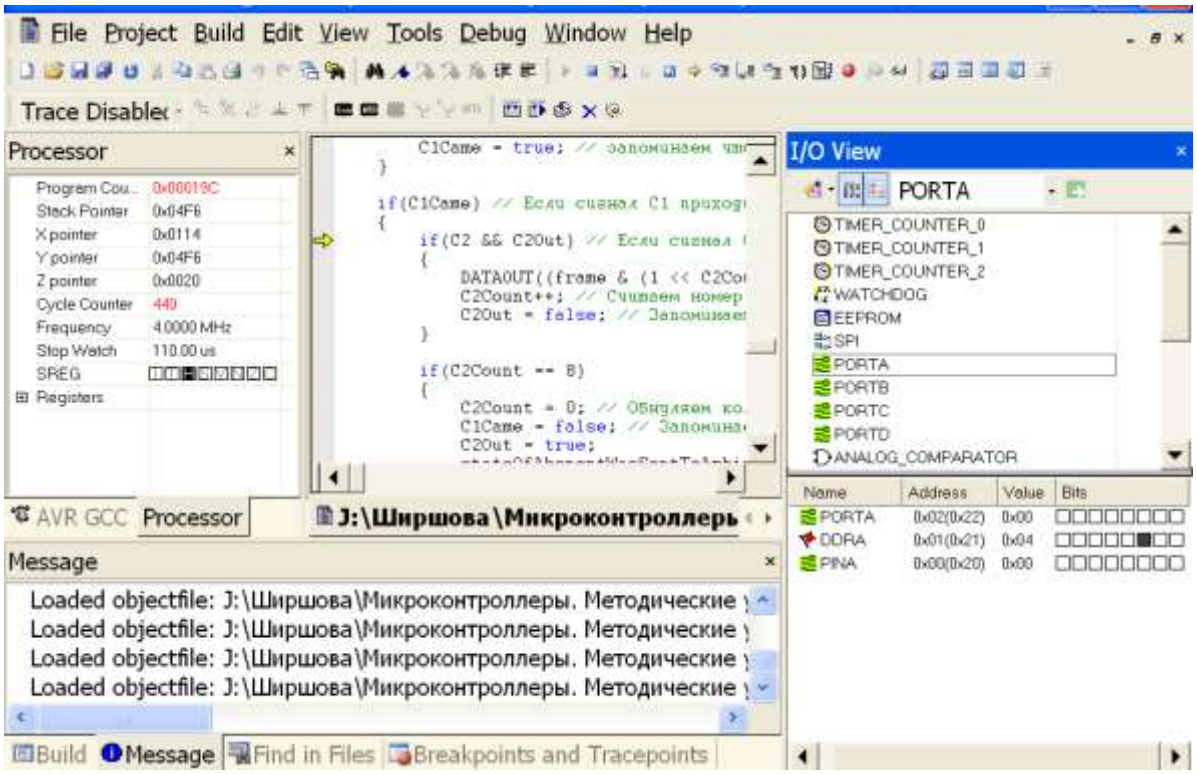
if(C1) // Если пришёл сигнал C 1,



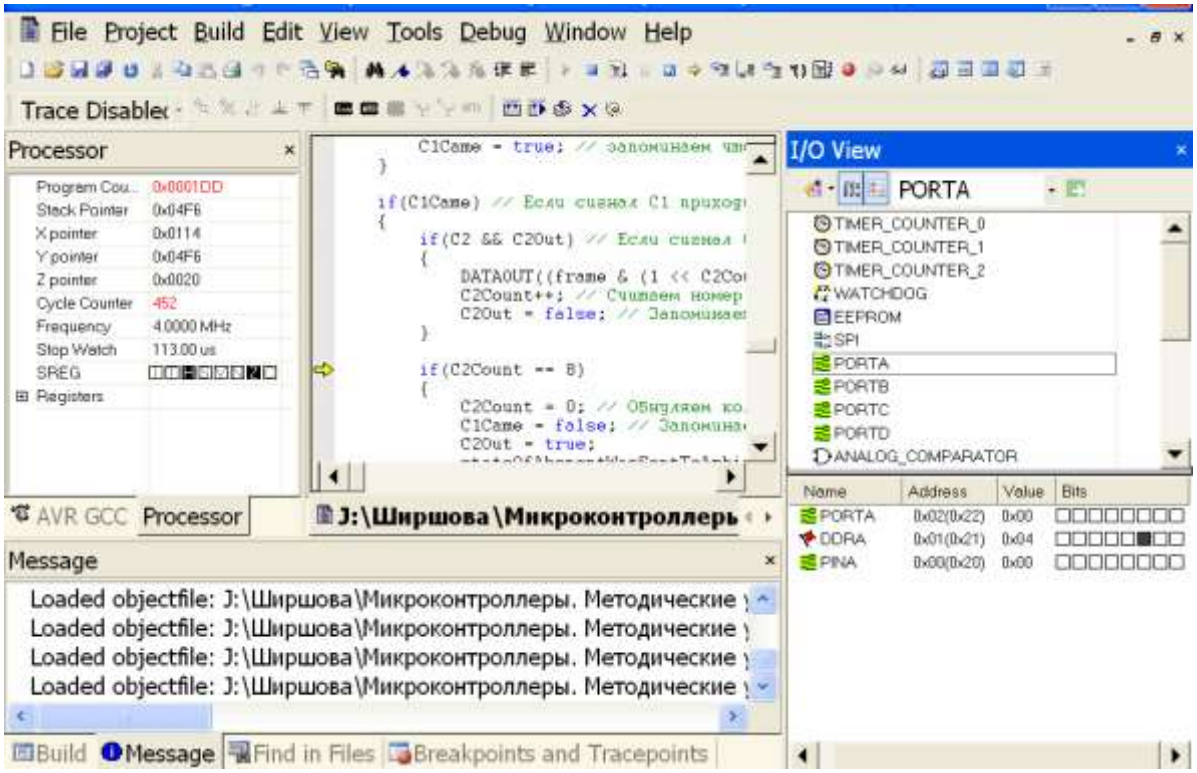
if(C1Came) // Если сигнал С 1 приходил



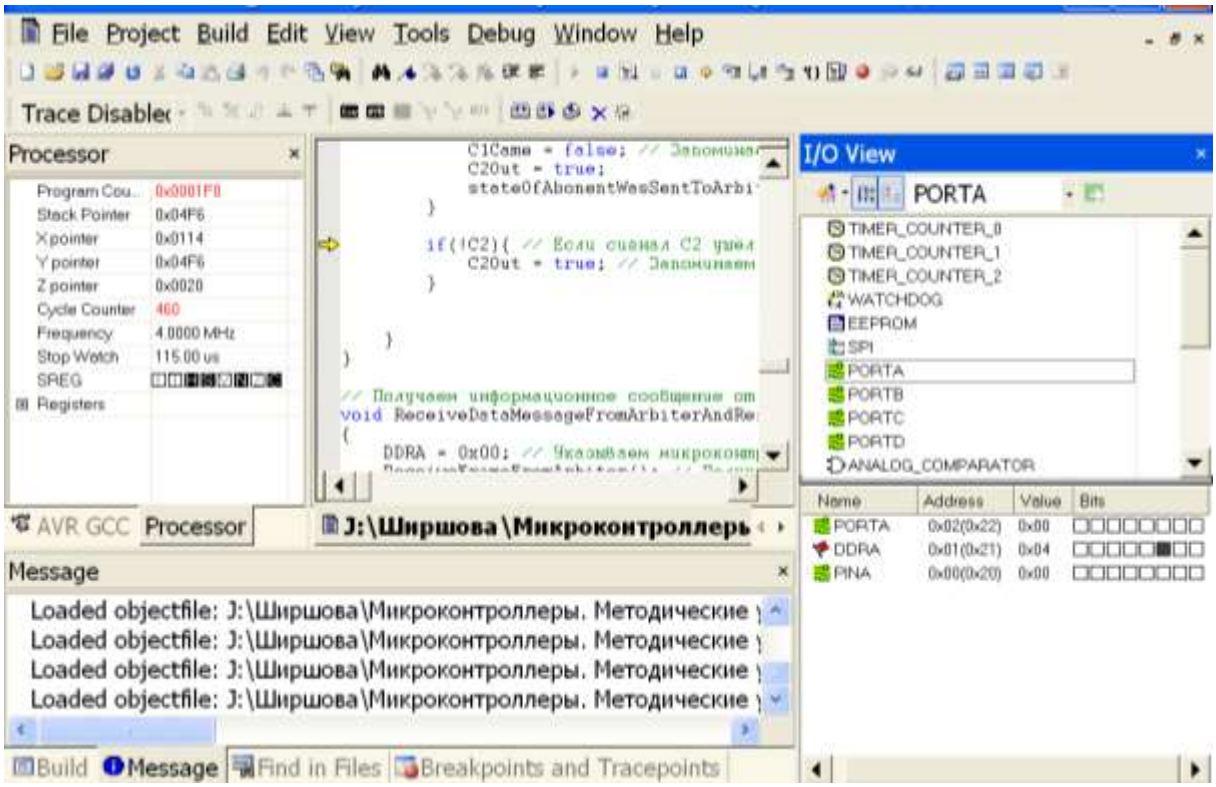
if(C2 && C2Out) // Если сигнал С 2 пришёл,



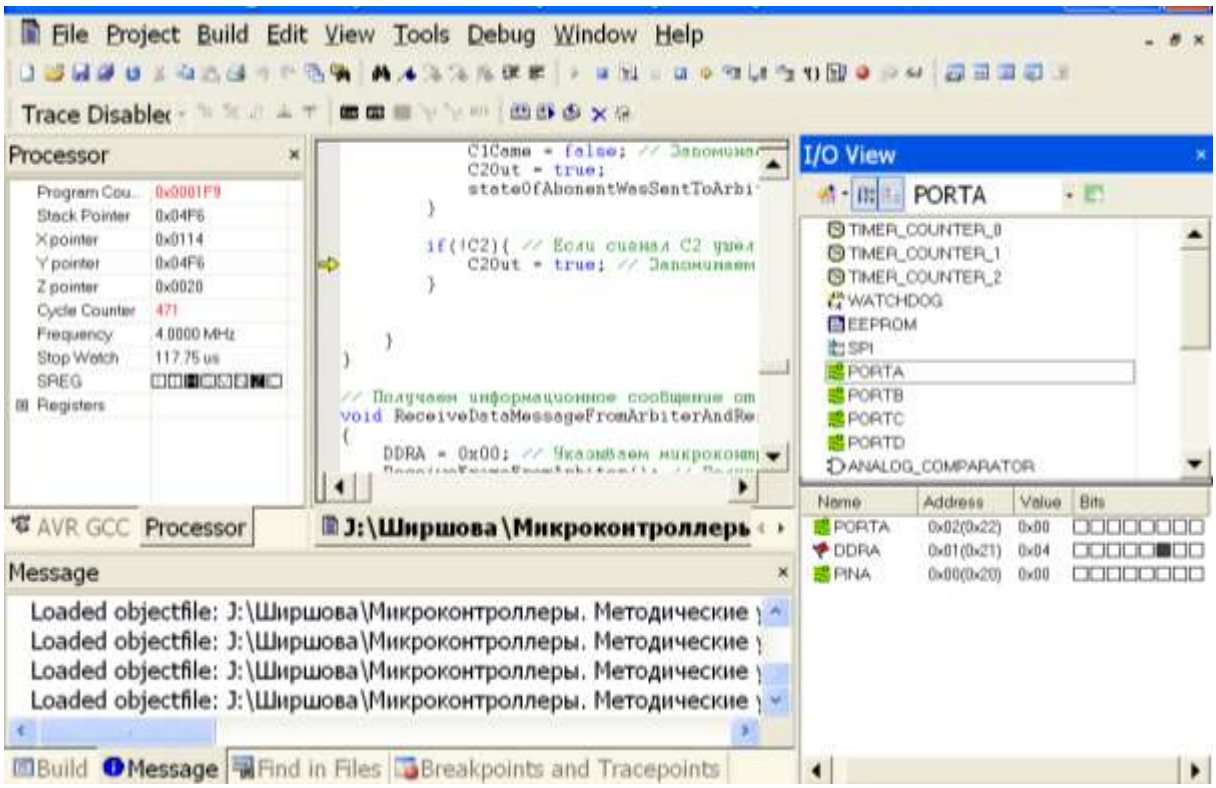
if(C2Count == 8)

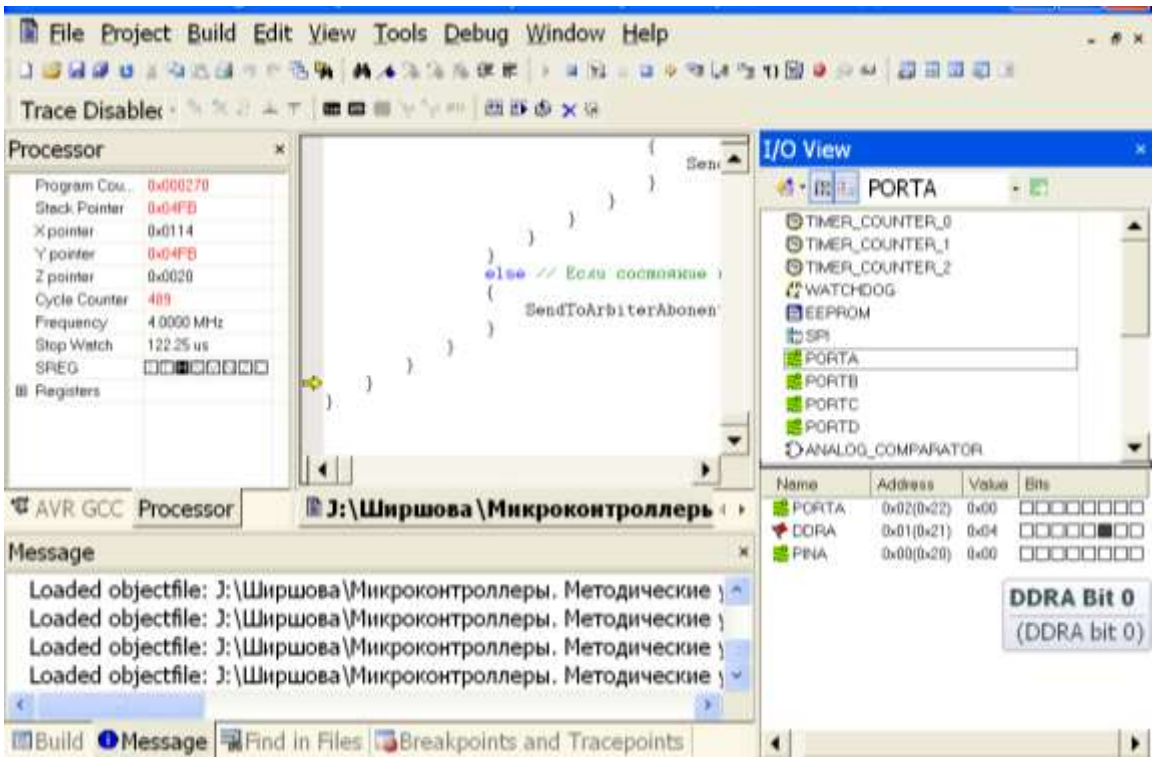


if(!C2){ // Если сигнал С 2 ушёл

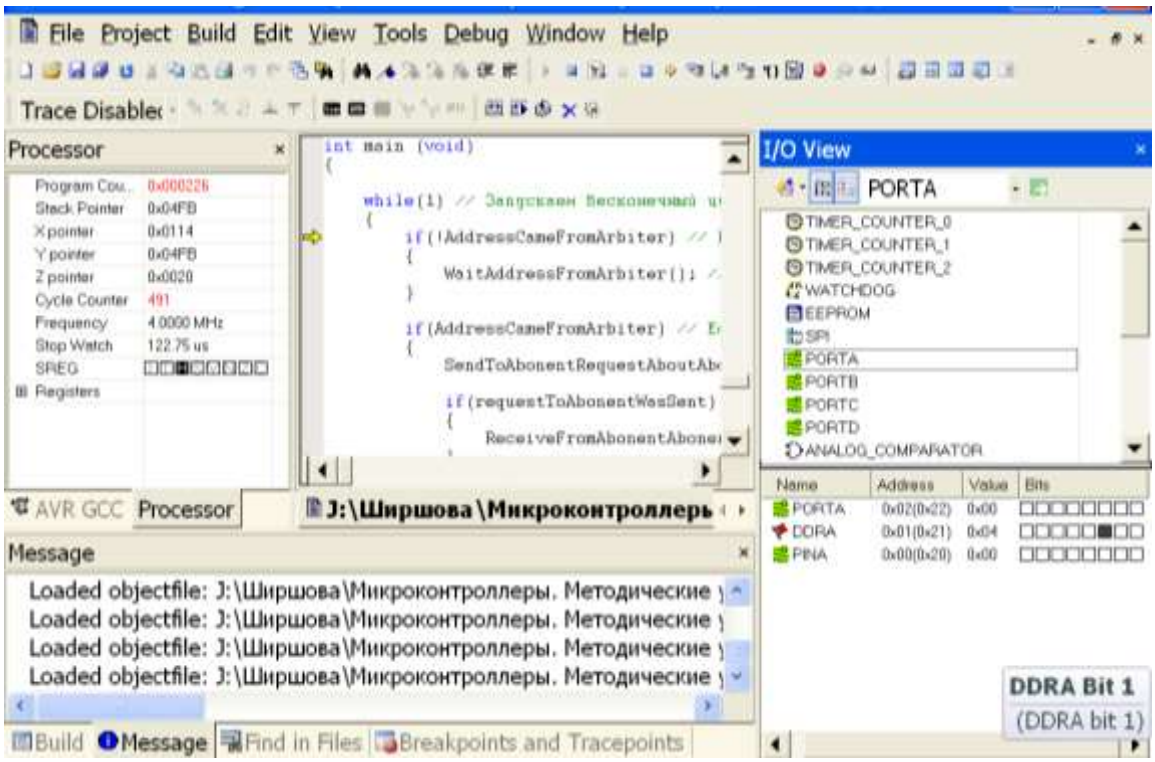


C2Out = true; // Запоминаем, что сигнал С 2 ушёл

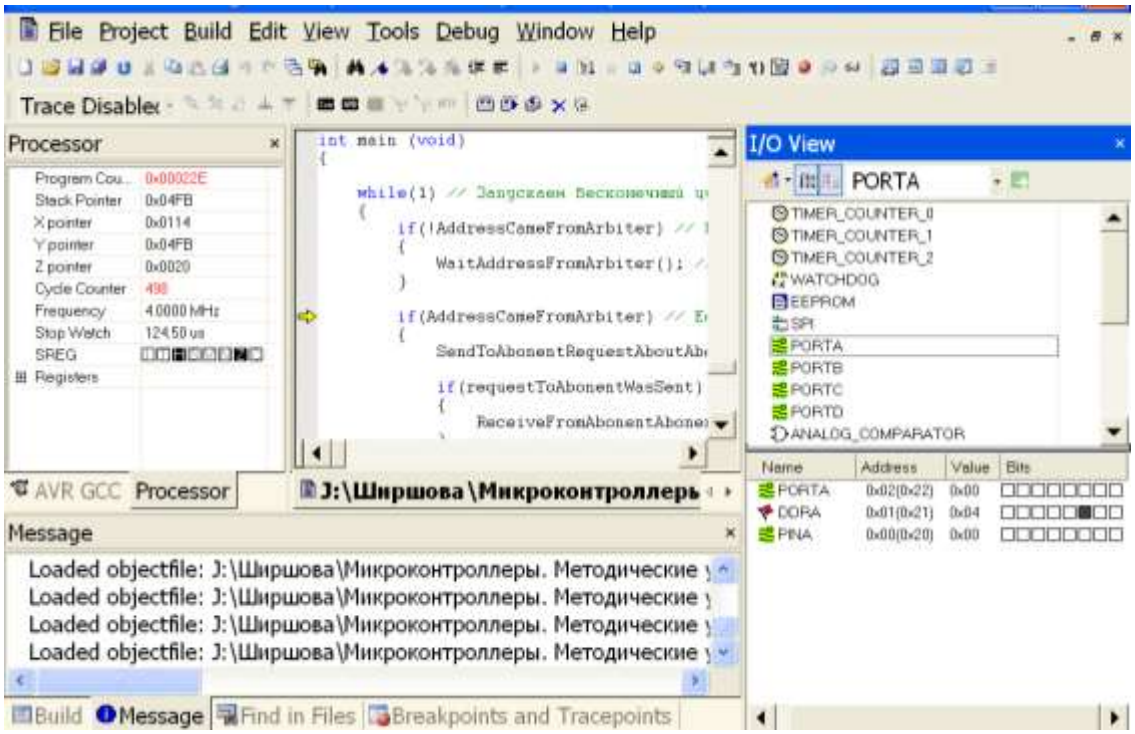




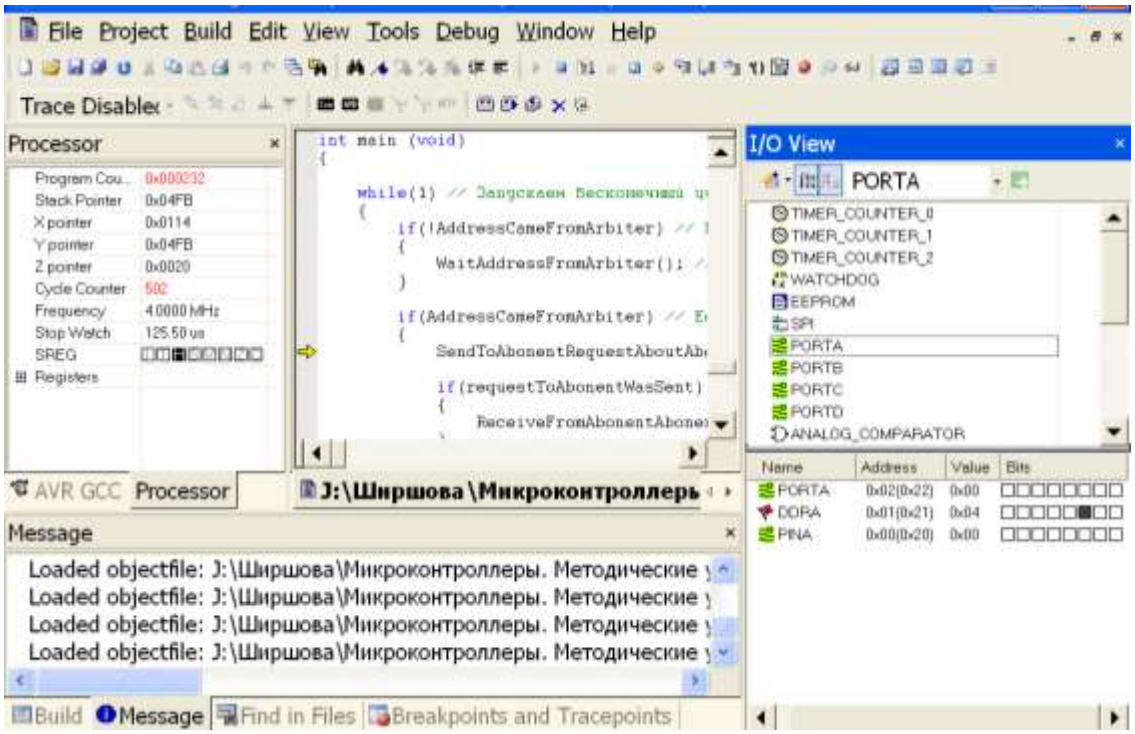
if(!AddressCameFromArbiter) // Если адрес МК ещё не пришёл от Арбитра,



if(AddressCameFromArbiter) // Если от Арбитра пришёл адрес МК

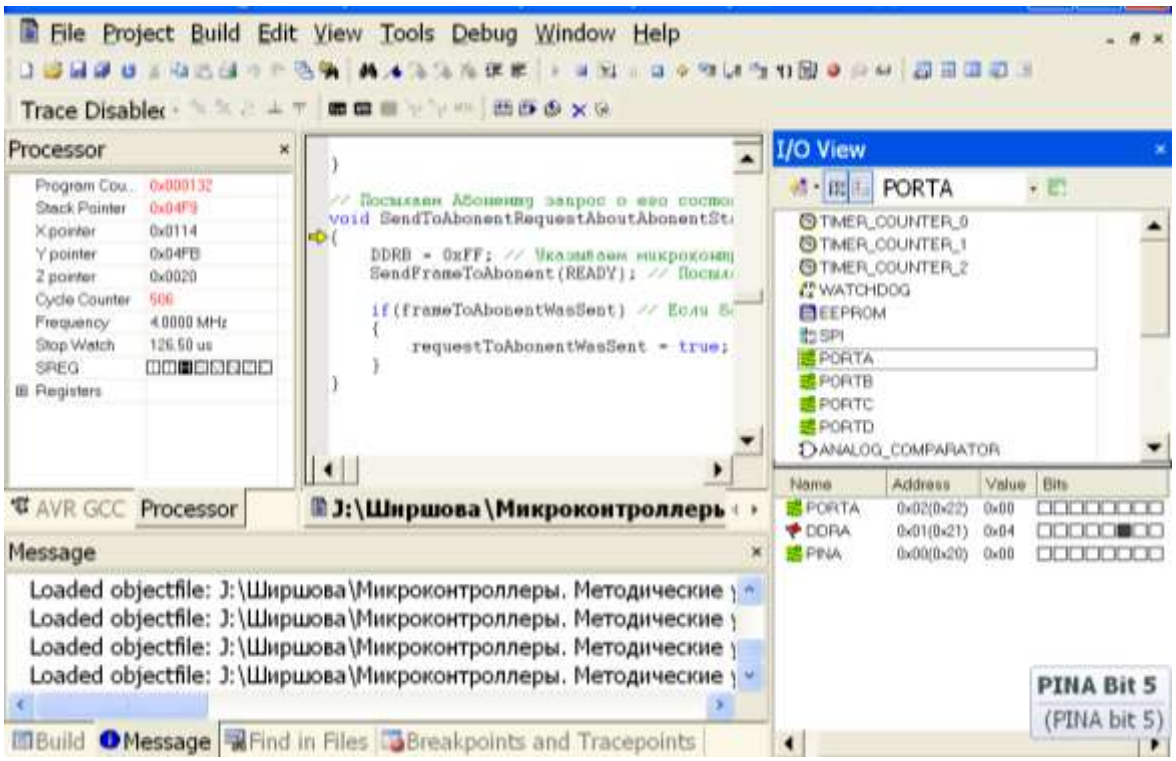


SendToAbonentRequestAboutAbonentState(); // посылаем Абоненту запрос о его состоянии

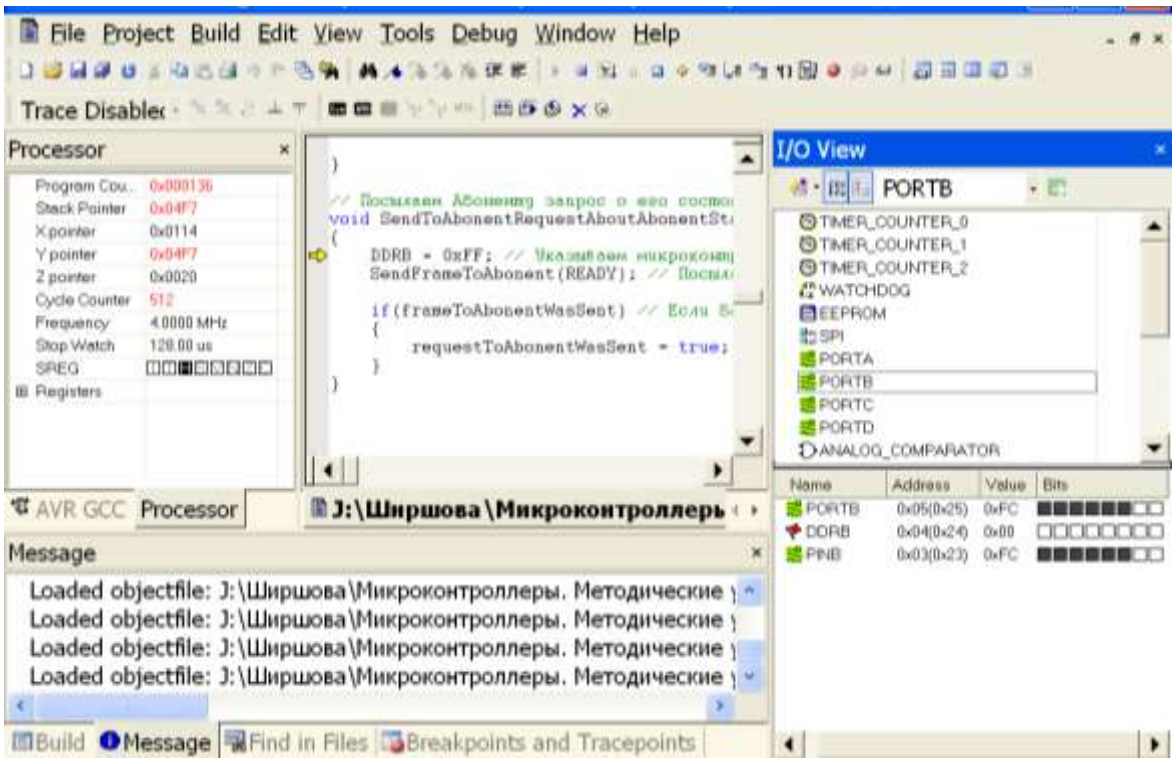


// Посылаем Абоненту запрос о его состоянии

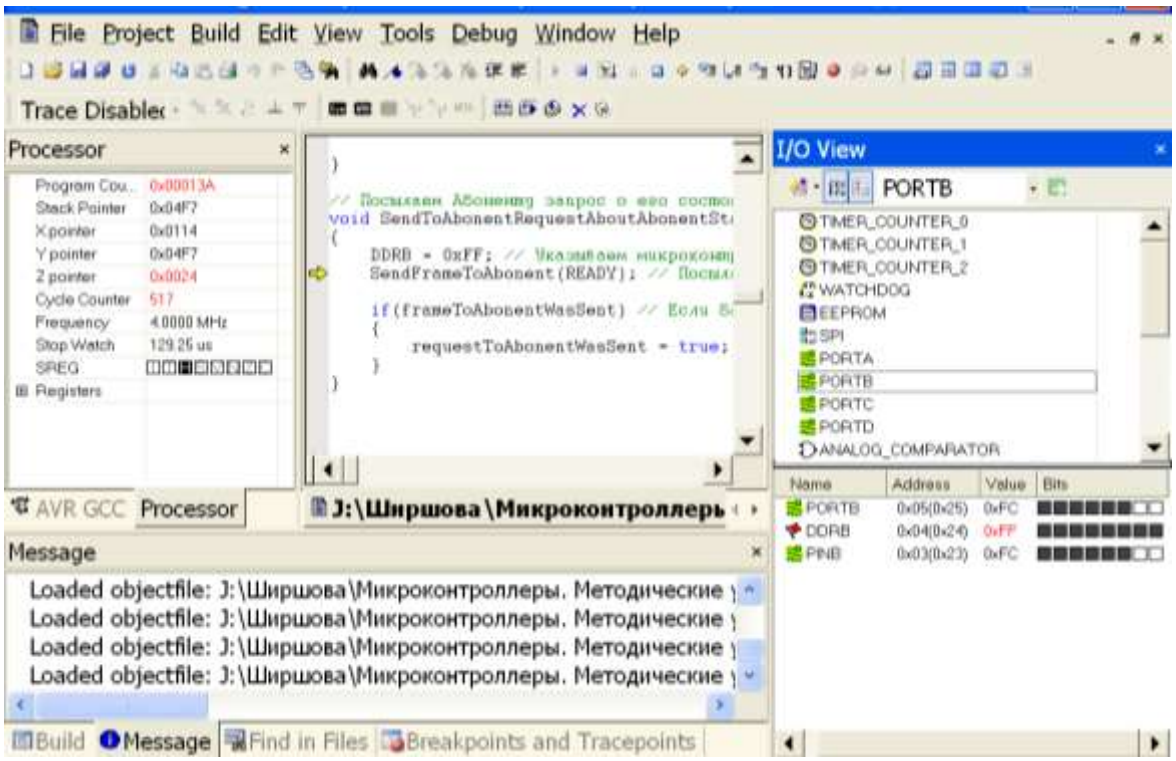
void SendToAbonentRequestAboutAbonentState()



DDRB = 0xFF; // Указываем микроконтроллеру настроить все выводы порта B на вывод информации

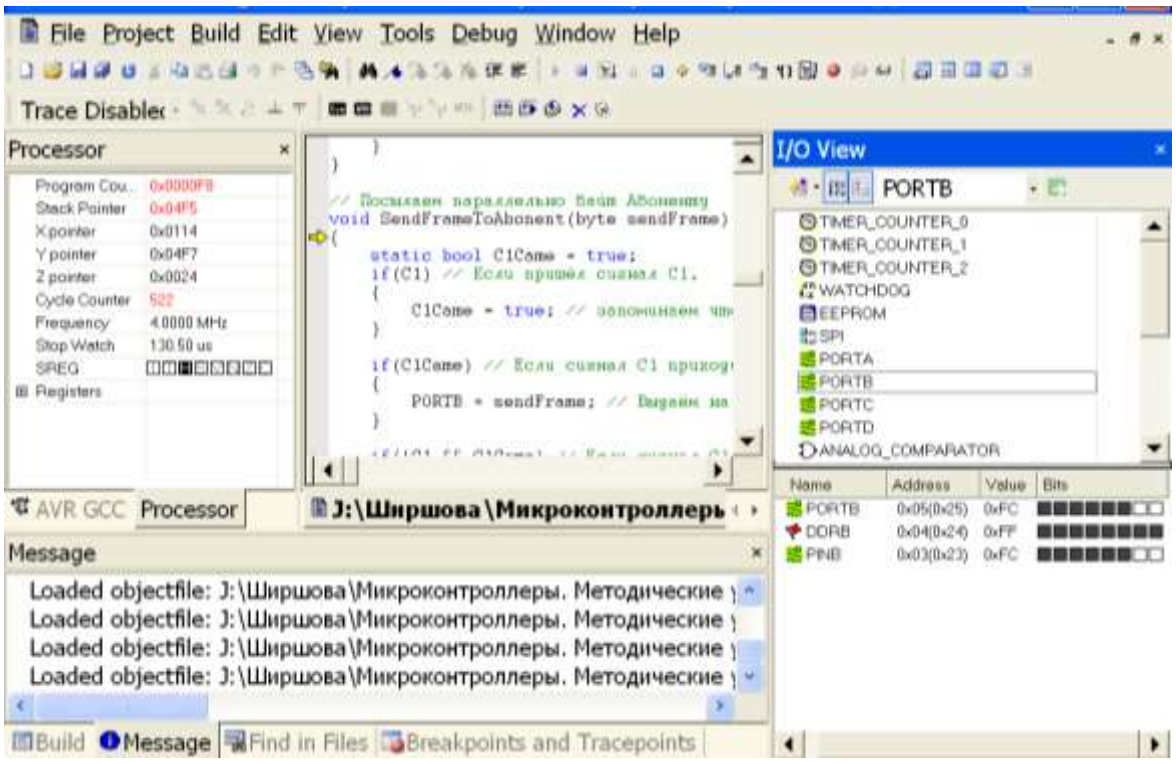


SendFrameToAbonent(READY); // Посылаем байт ГОТОВ Абоненту

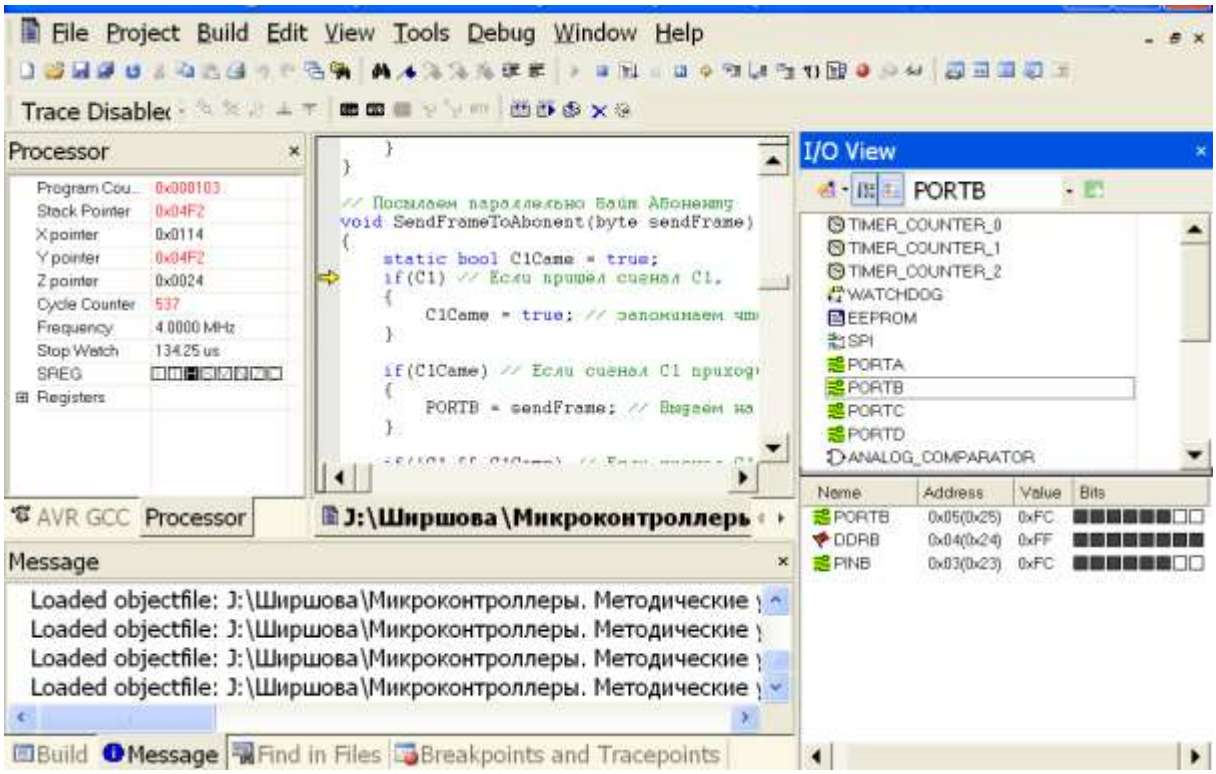


// Посылаем параллельно байт Абоненту

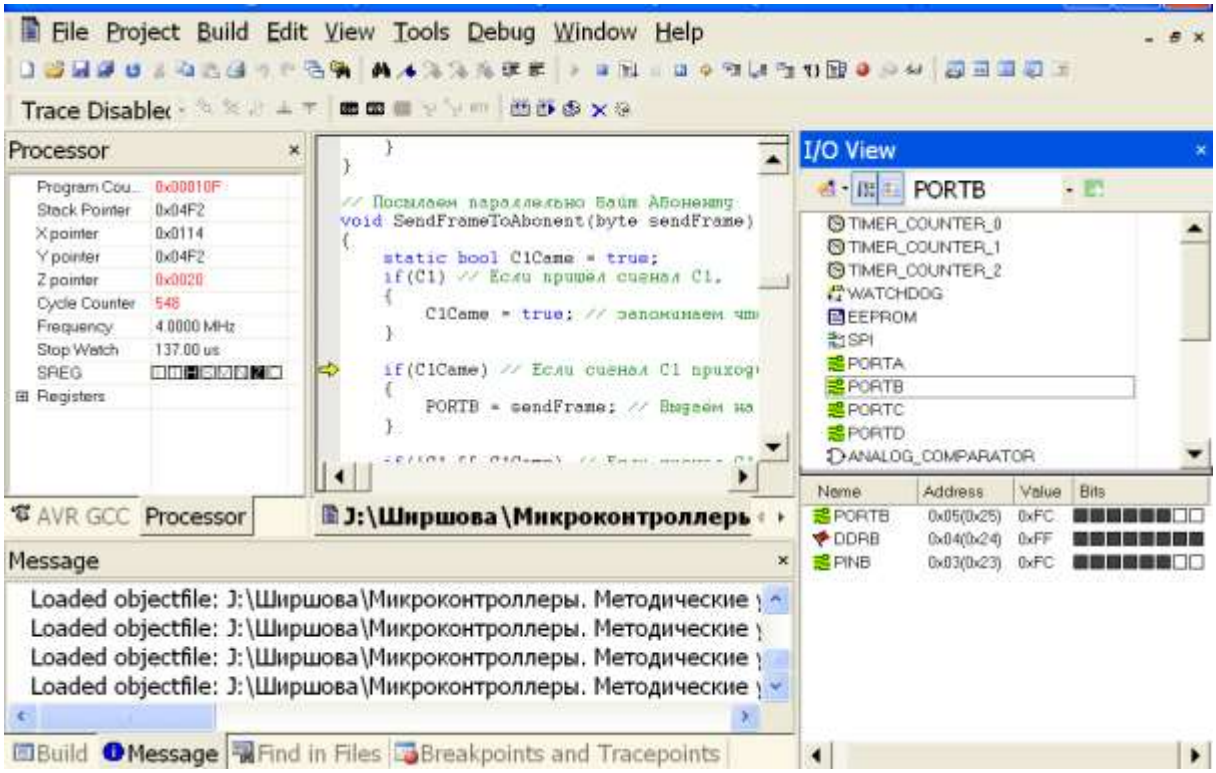
void SendFrameToAbonent(byte sendFrame)



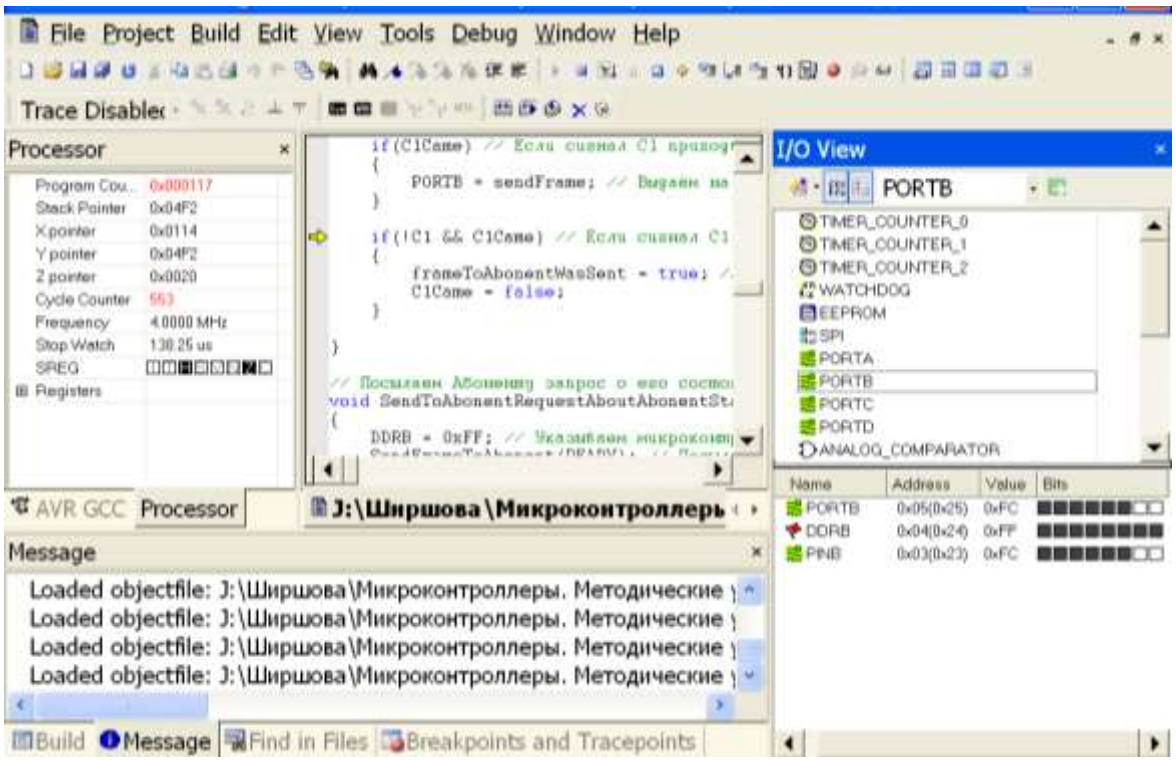
if(C1) // Если пришёл сигнал С 1,



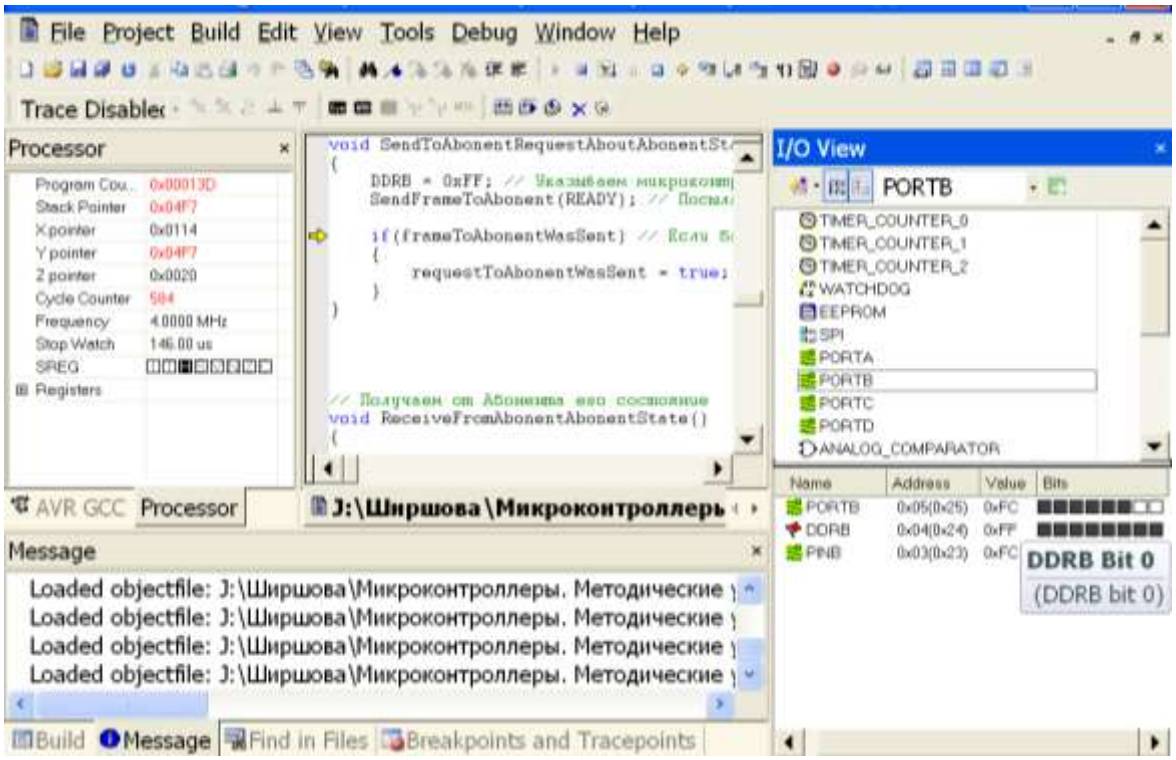
if(C1Came) // Если сигнал С 1 приходил



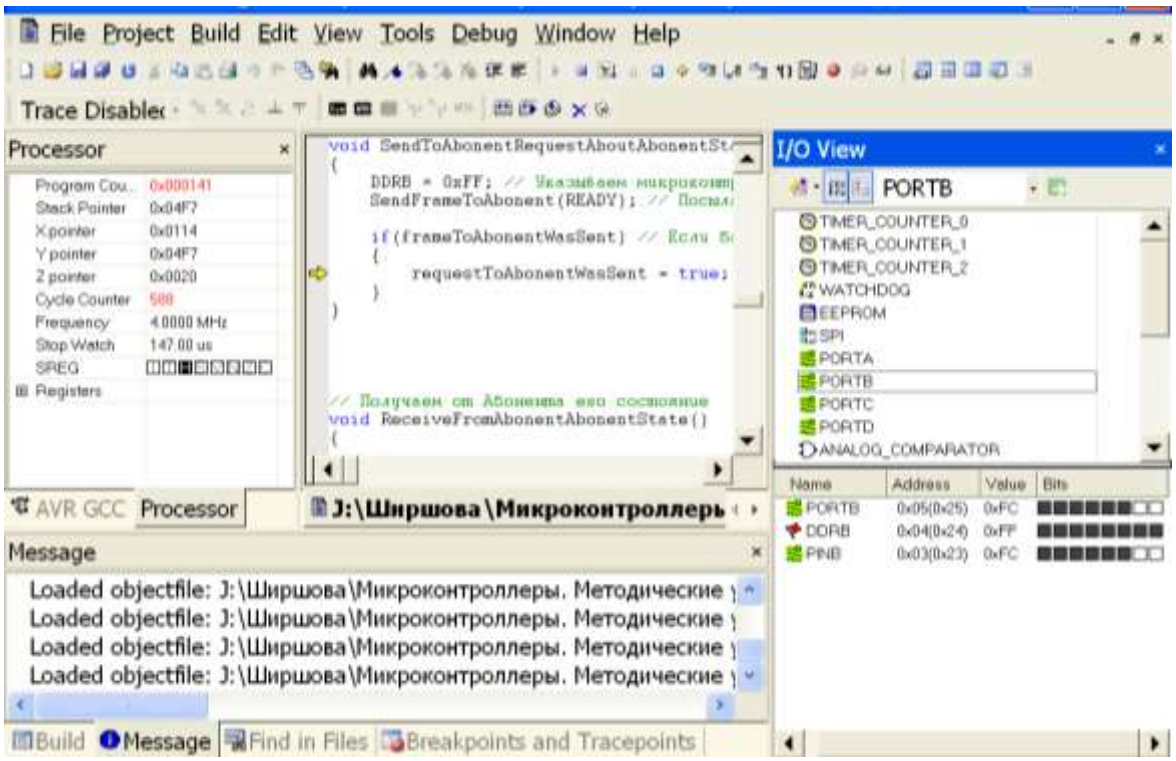
if(!C1 && C1Came) // Если сигнал С 1 ушёл и приходил



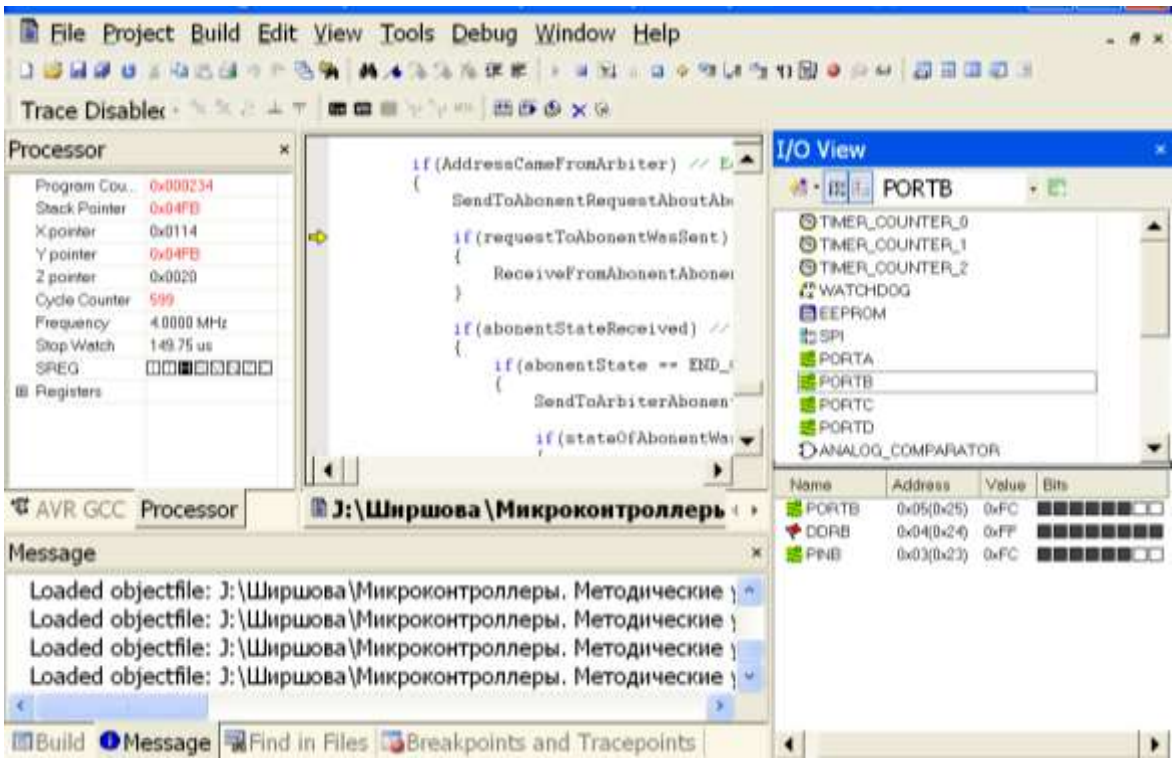
if(frameToAbonentWasSent) // Если байт был послан



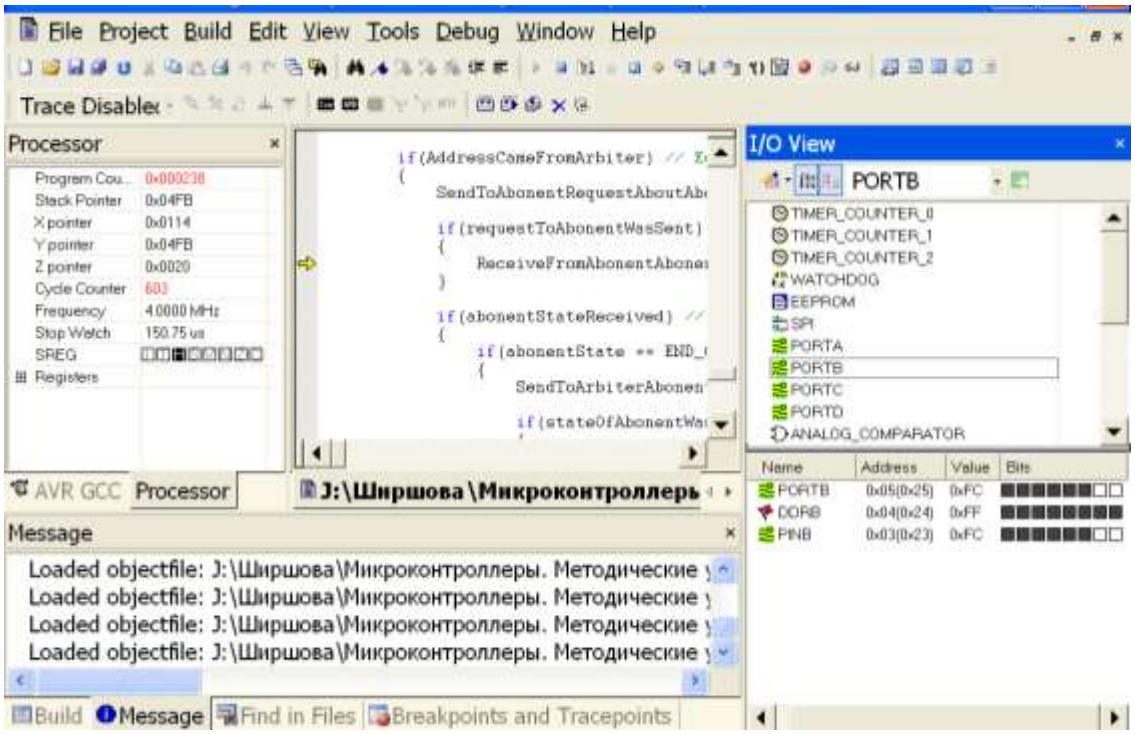
requestToAbonentWasSent = true; // Запоминаем, что запрос Абоненту был послан



if(requestToAbonentWasSent) // Если запрос о состоянии Абонента был послан

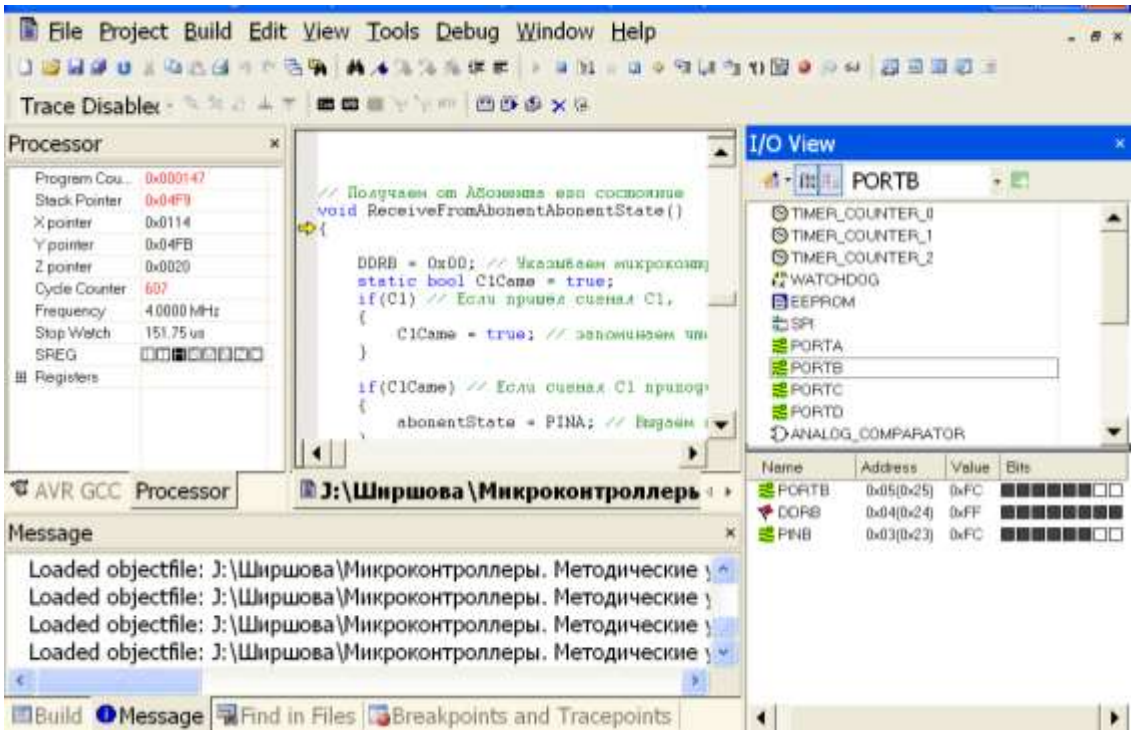


ReceiveFromAbonentAbonentState(); // получаем ответ

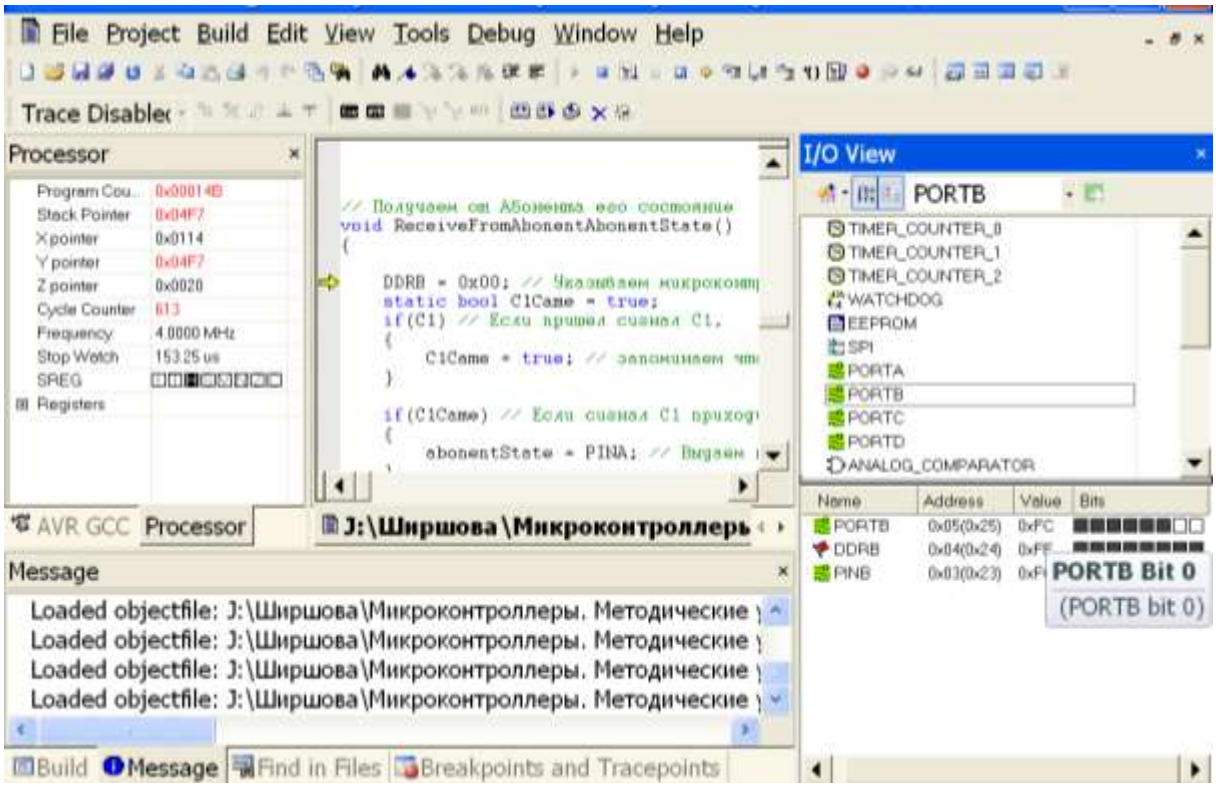


// Получаем от Абонента его состояние

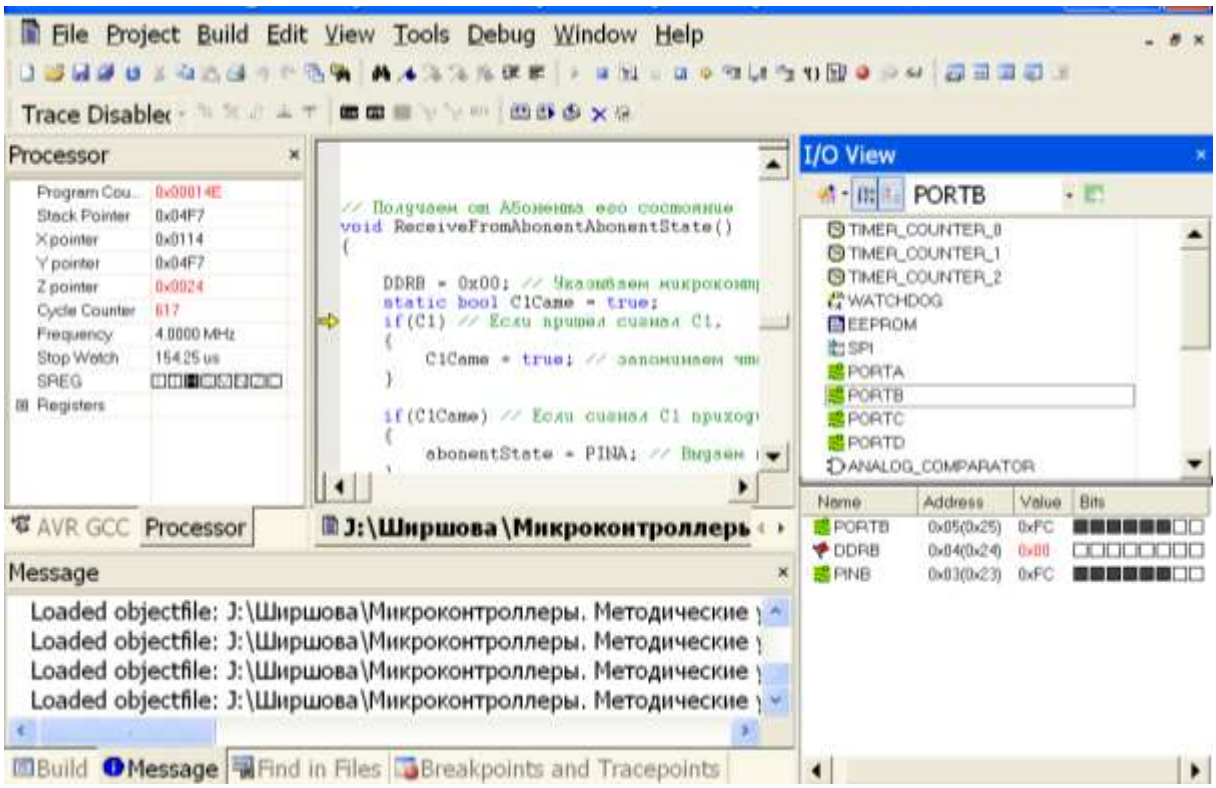
void ReceiveFromAbonentAbonentState()



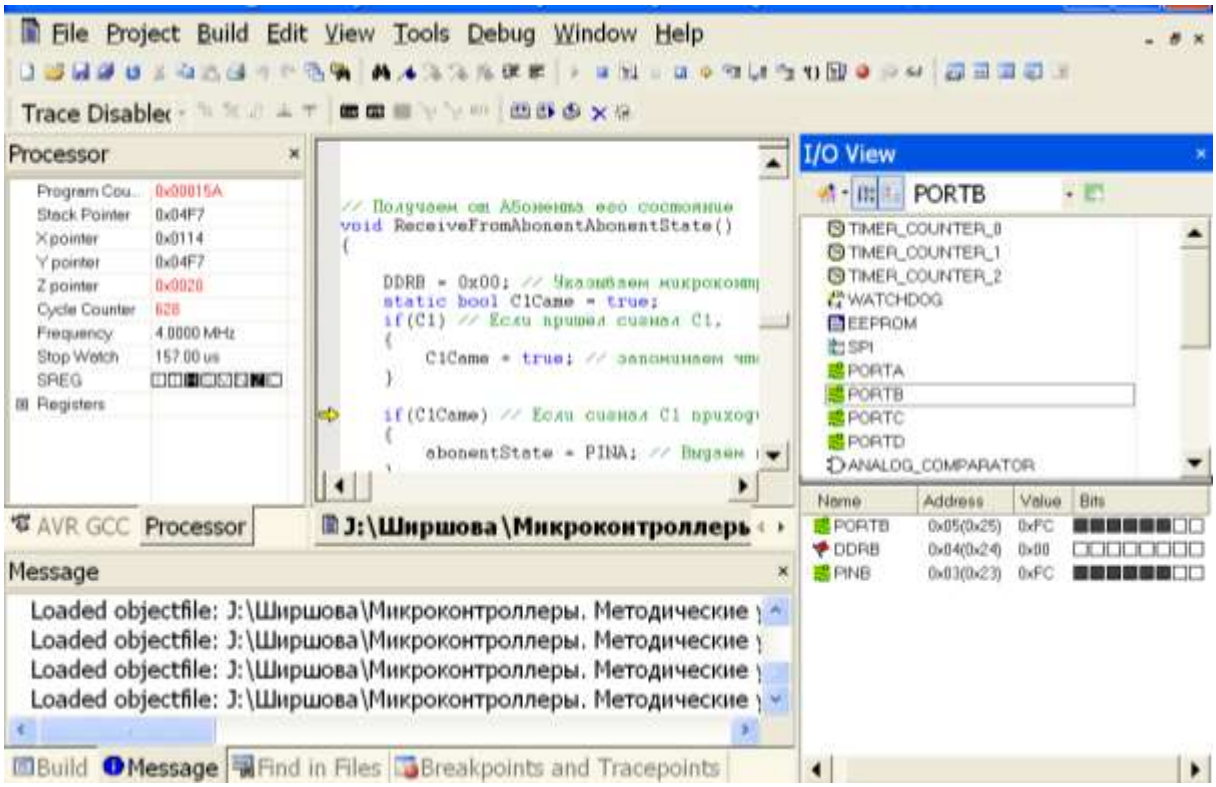
DDRB = 0x00; // Указываем микроконтроллеру настроить все выводы порта B на приём информации



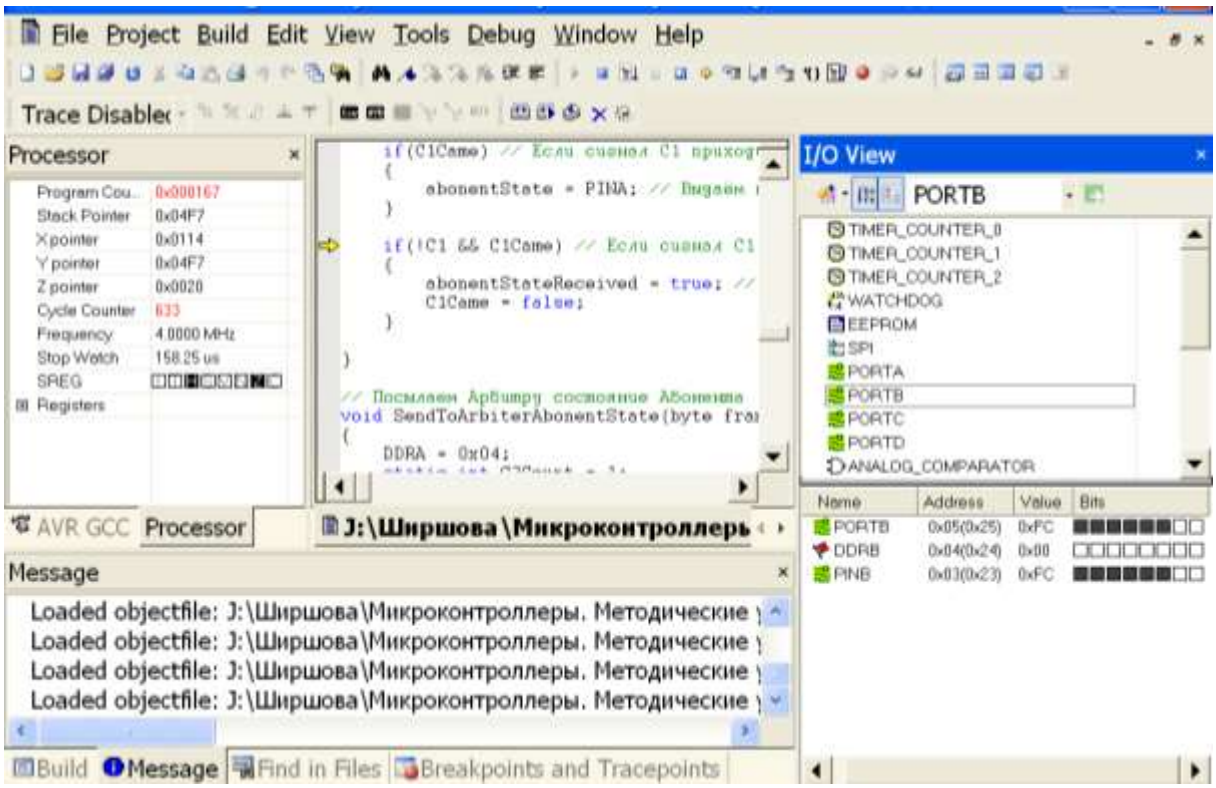
if(C1) // Если пришёл сигнал С 1,



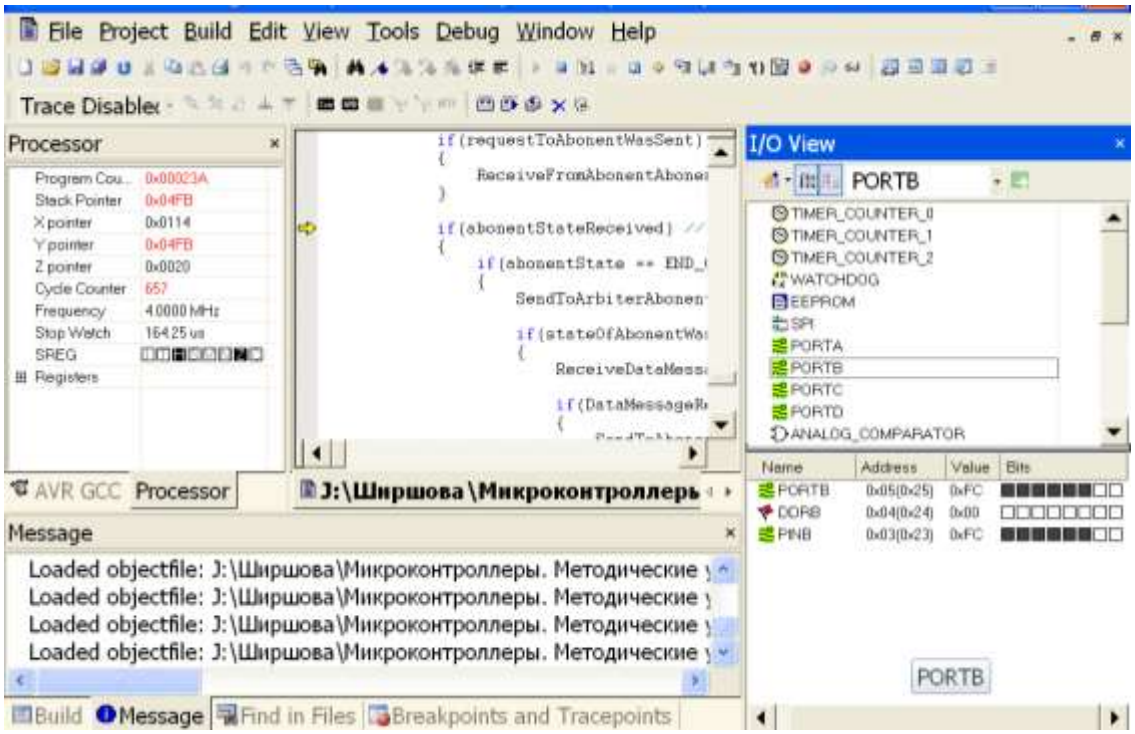
if(C1Came) // Если сигнал С 1 приходил



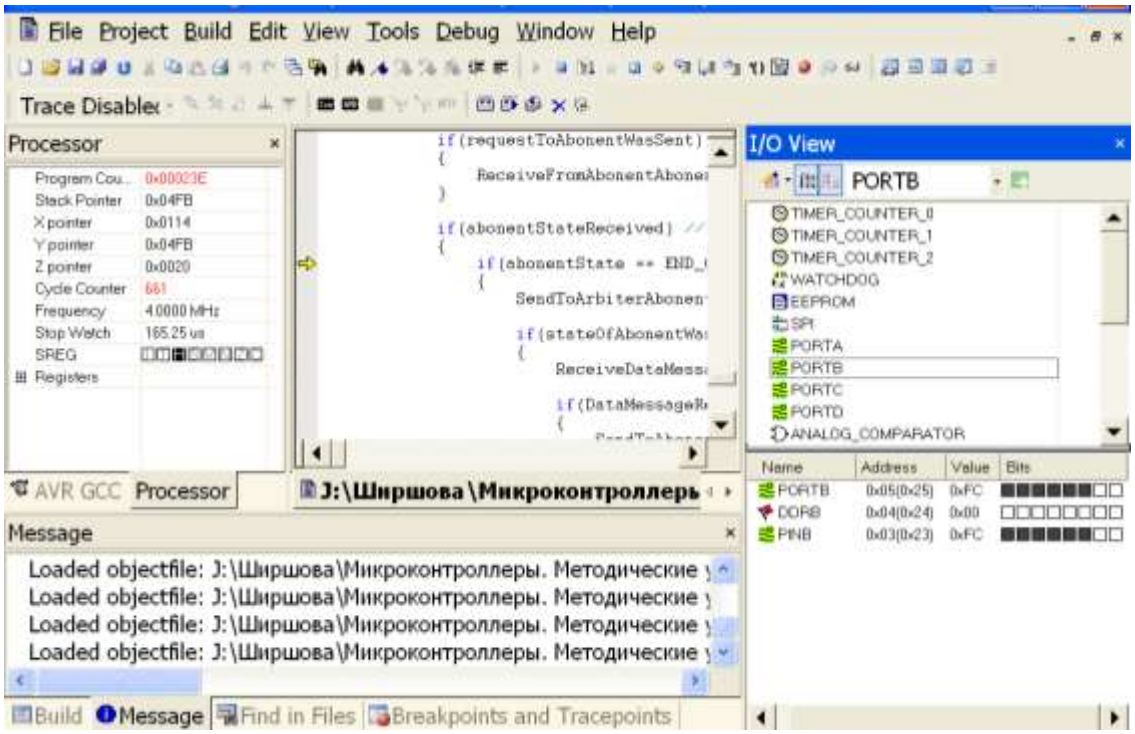
if(!C1 && C1Came) // Если сигнал С 1 ушёл и приходил



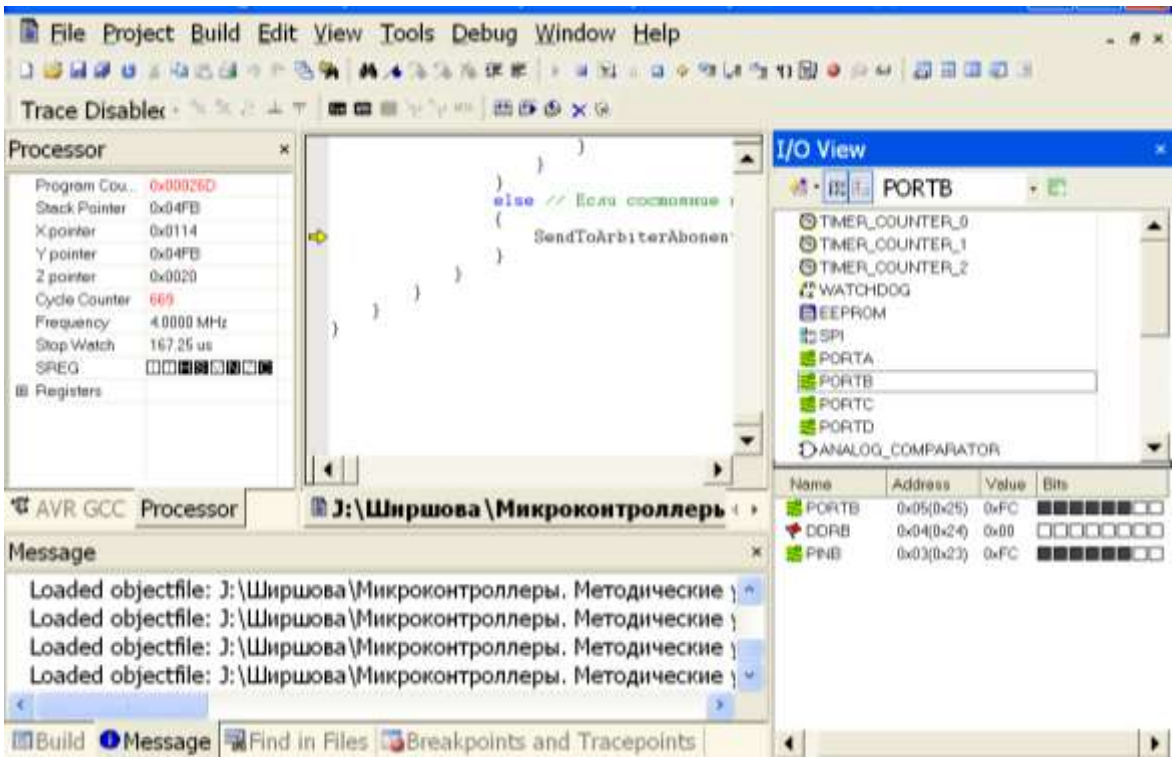
if(abonentStateReceived) // Если получили состояние Абонента,



if(abonentState == END_OF_WORK) // и состояние равно КОНЕЦ РАБОТЫ

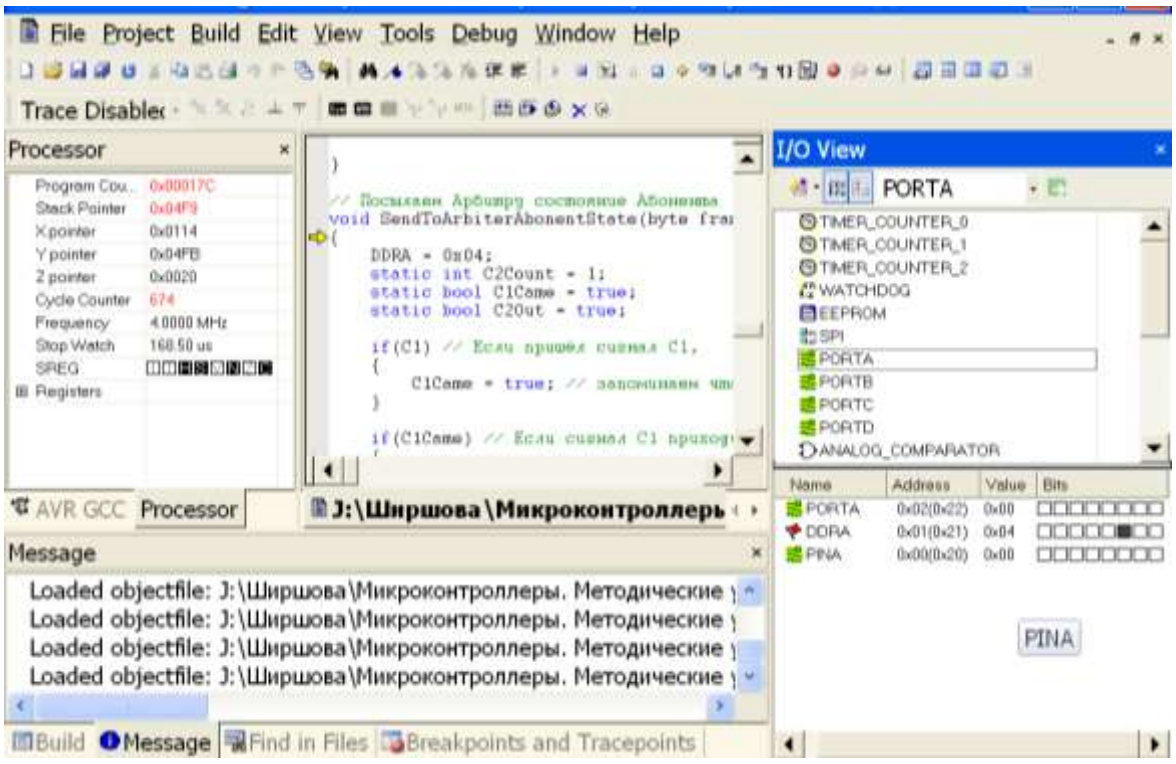


SendToArbiterAbonentState(BUSY); // Посылаем Арбитру, что Абонент ЗАНЯТ

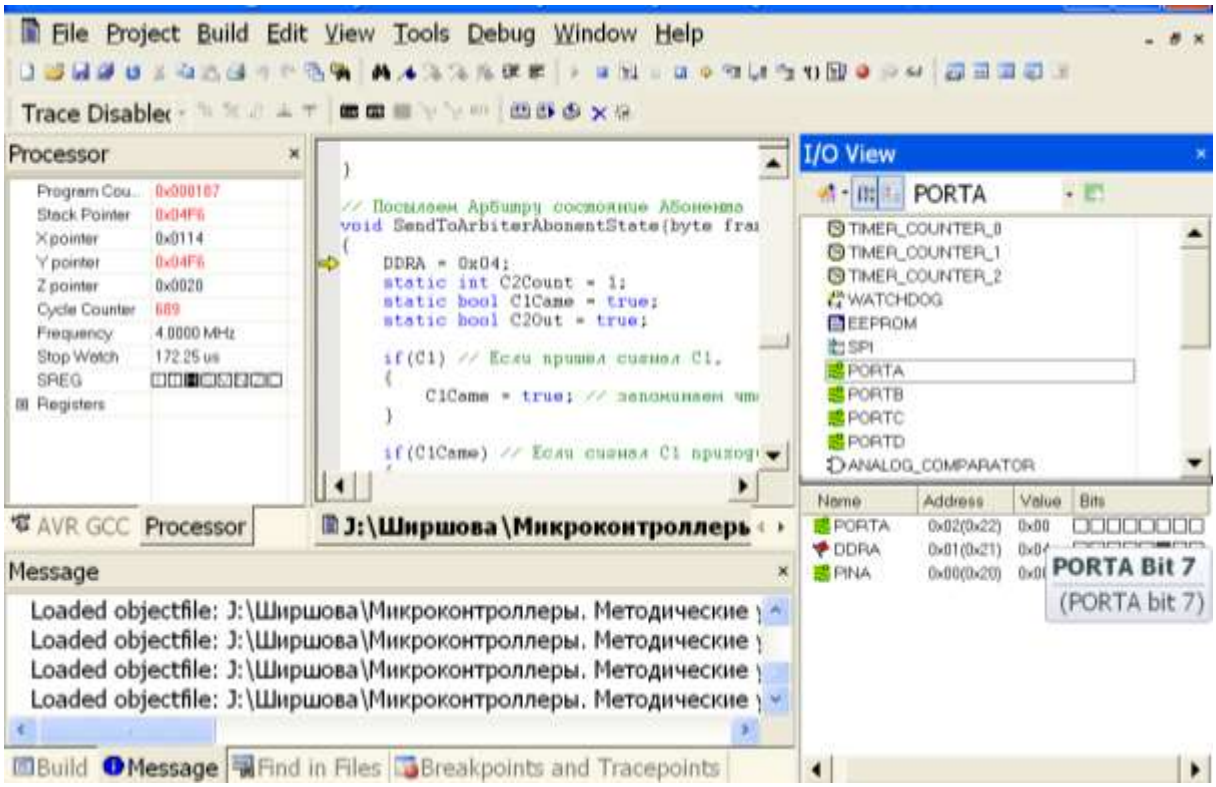


// Посылаем Арбитру состояние Абонента

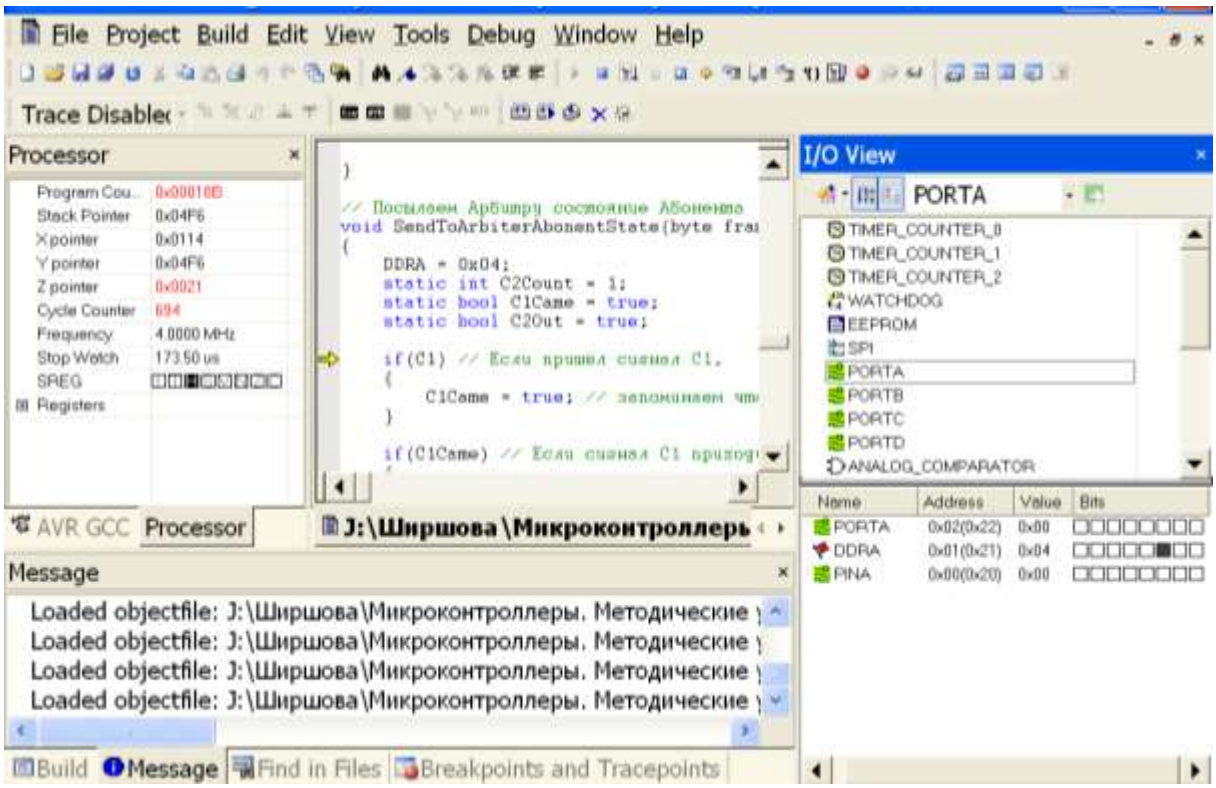
void SendToArbiterAbonentState(byte frame)



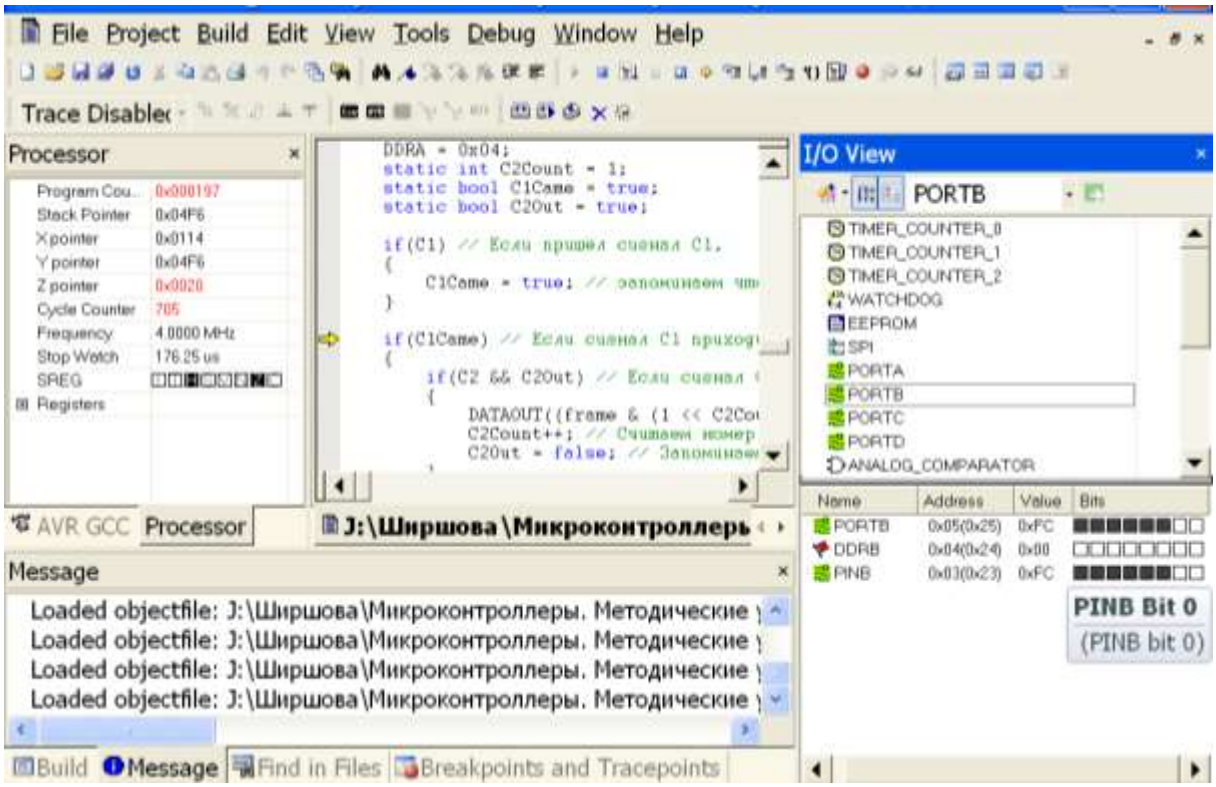
DDRA = 0x04;



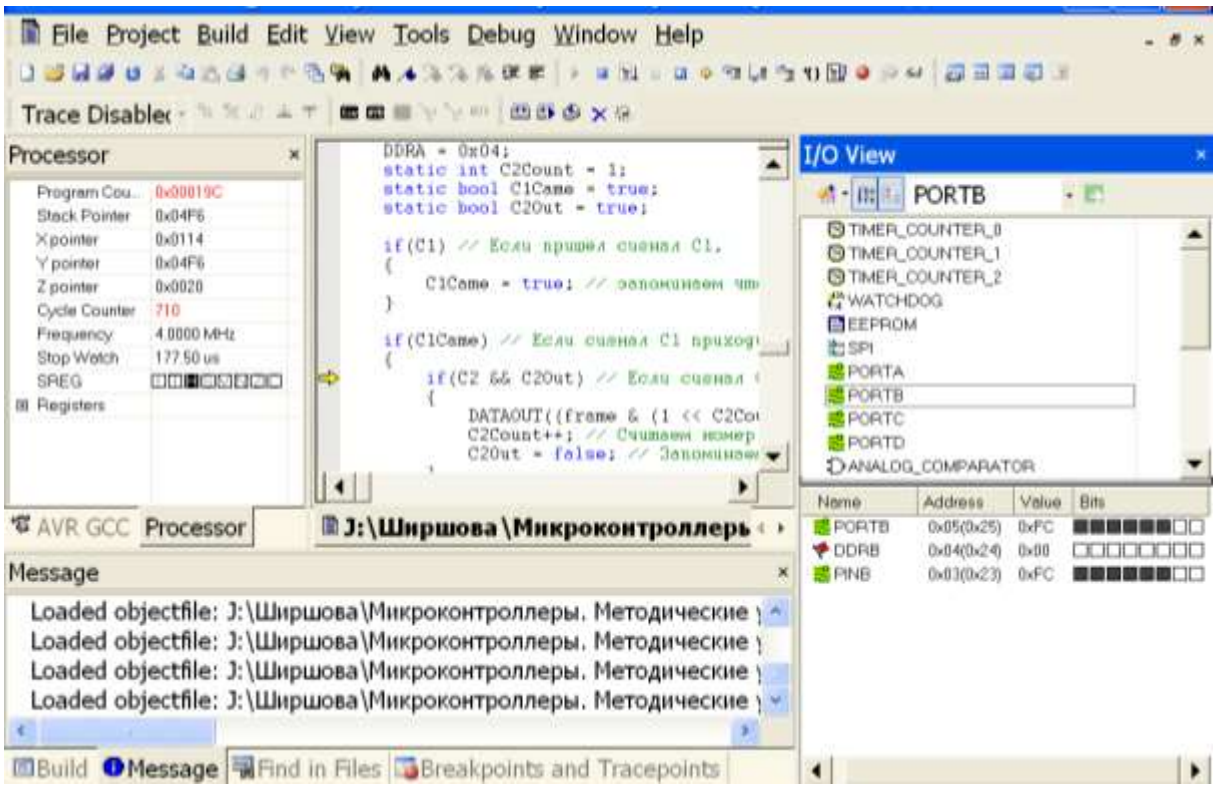
if(C1) // Если пришёл сигнал С 1,



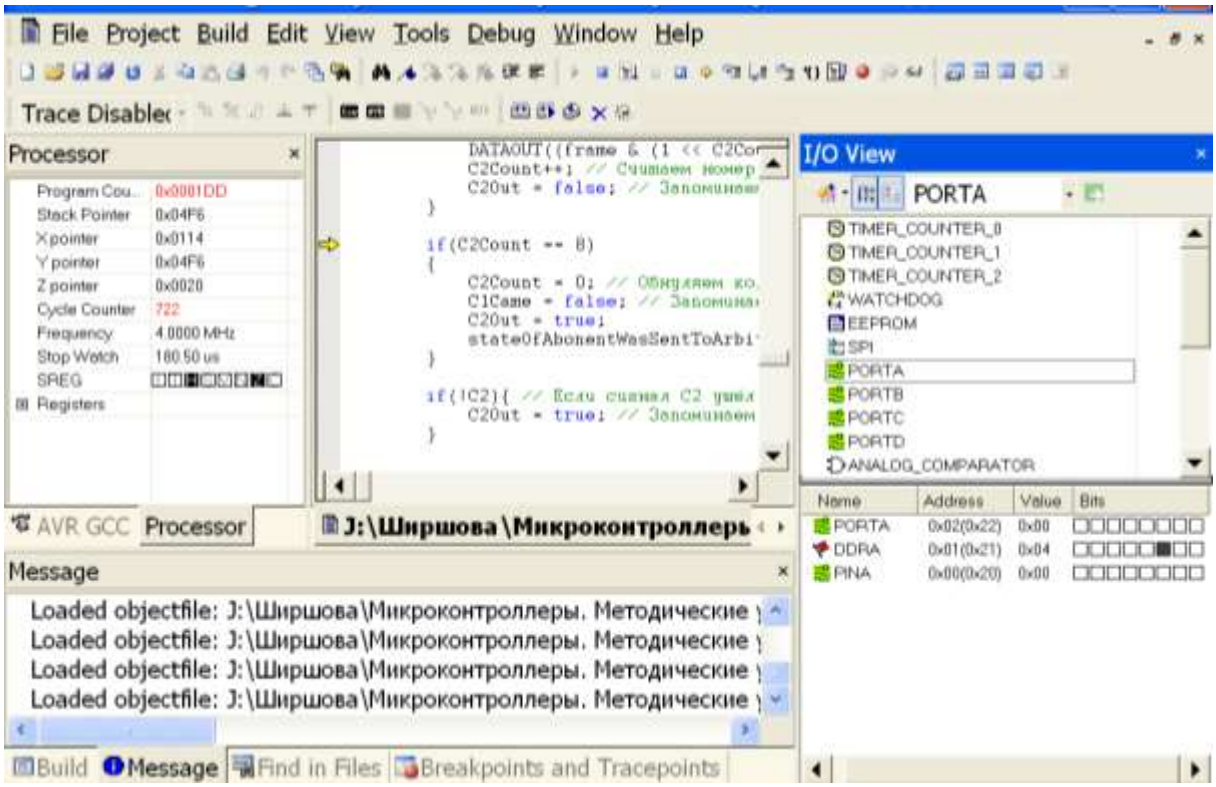
if(C1Came) // Если сигнал С 1 приходил



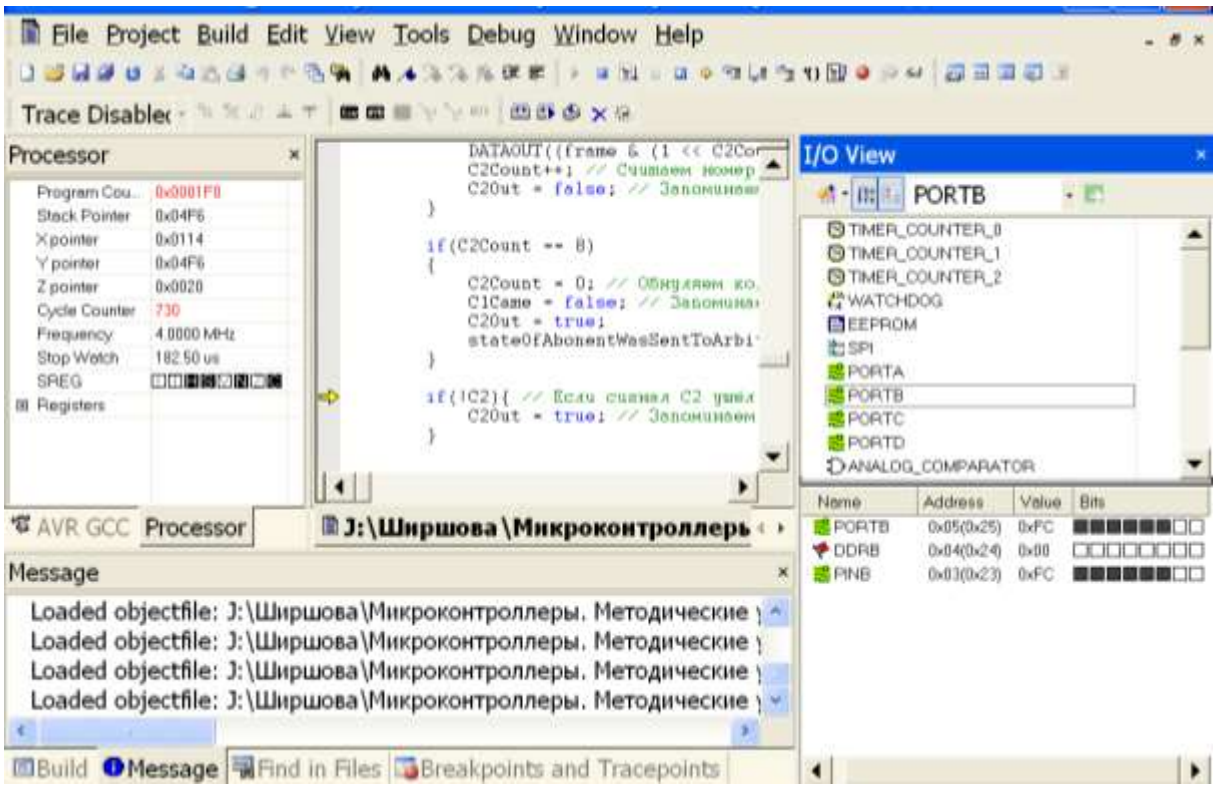
if(C2 && C2Out) // Если сигнал С 2 пришёл,



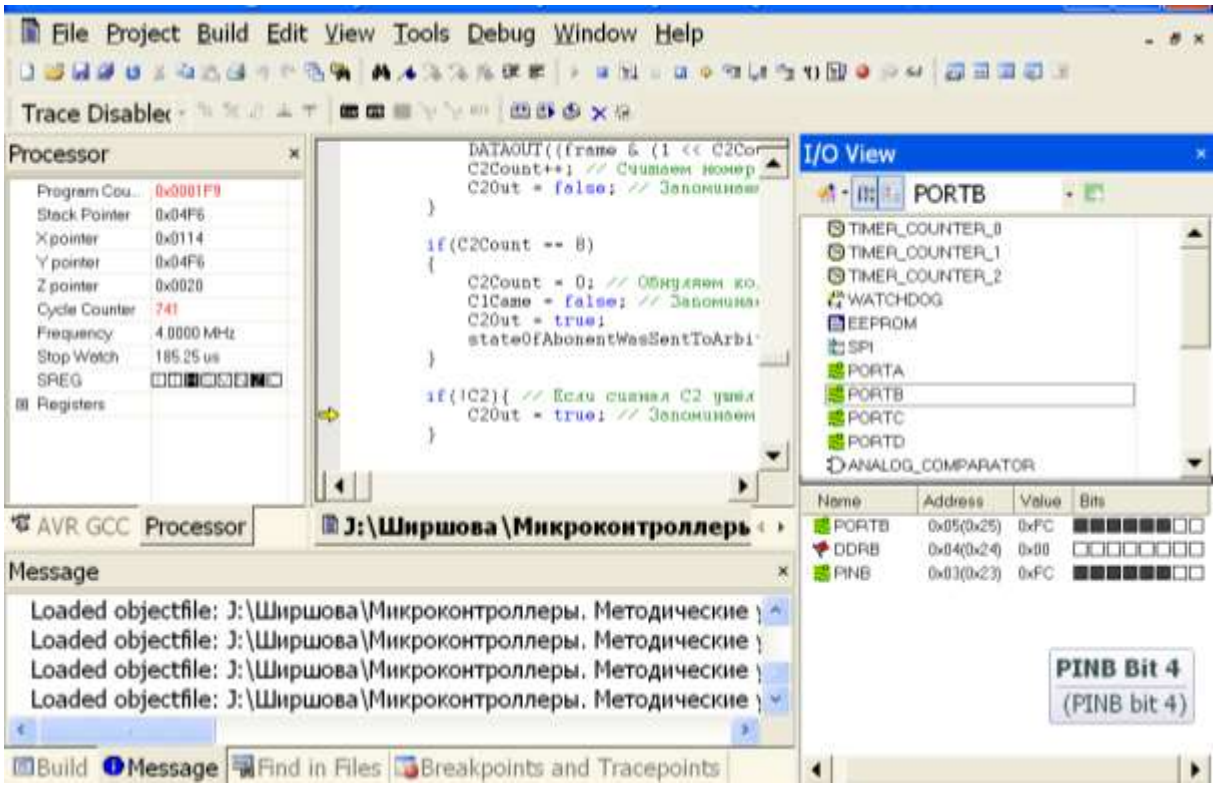
if(C2Count == 8)

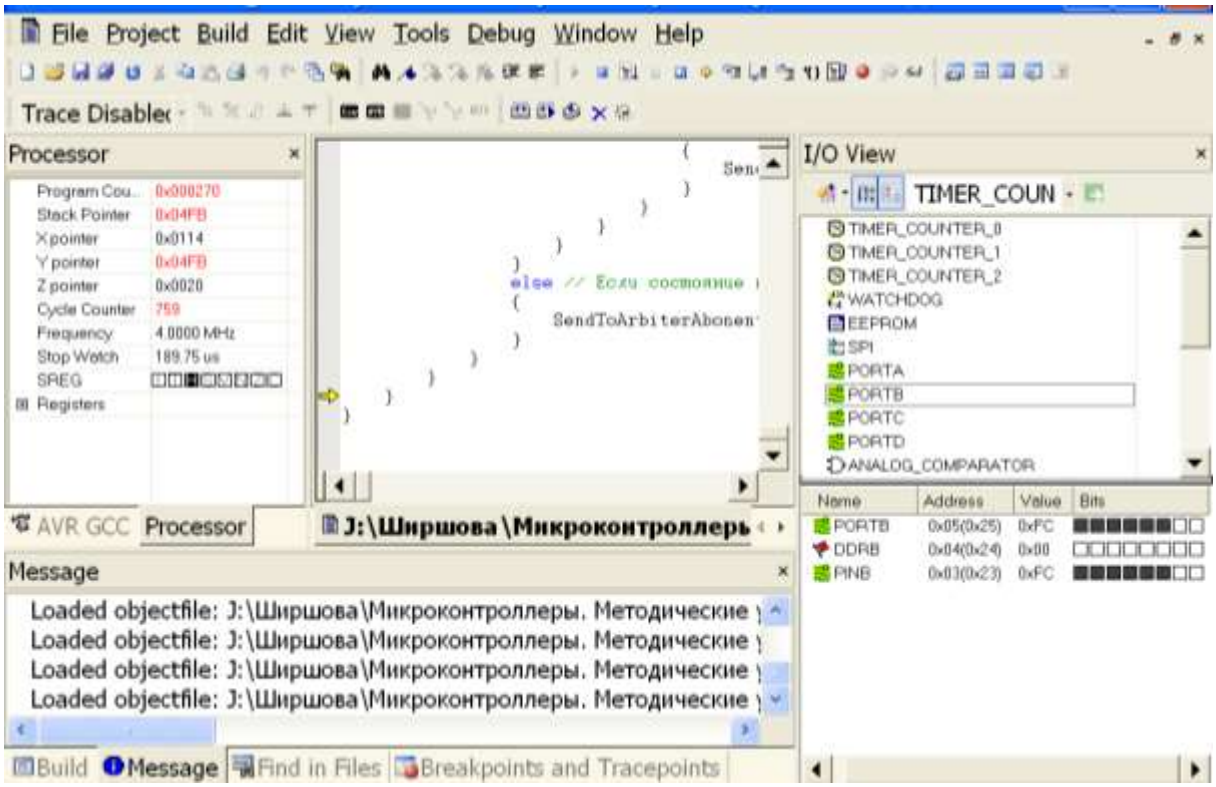


if(!C2){ // Если сигнал С 2 ушёл

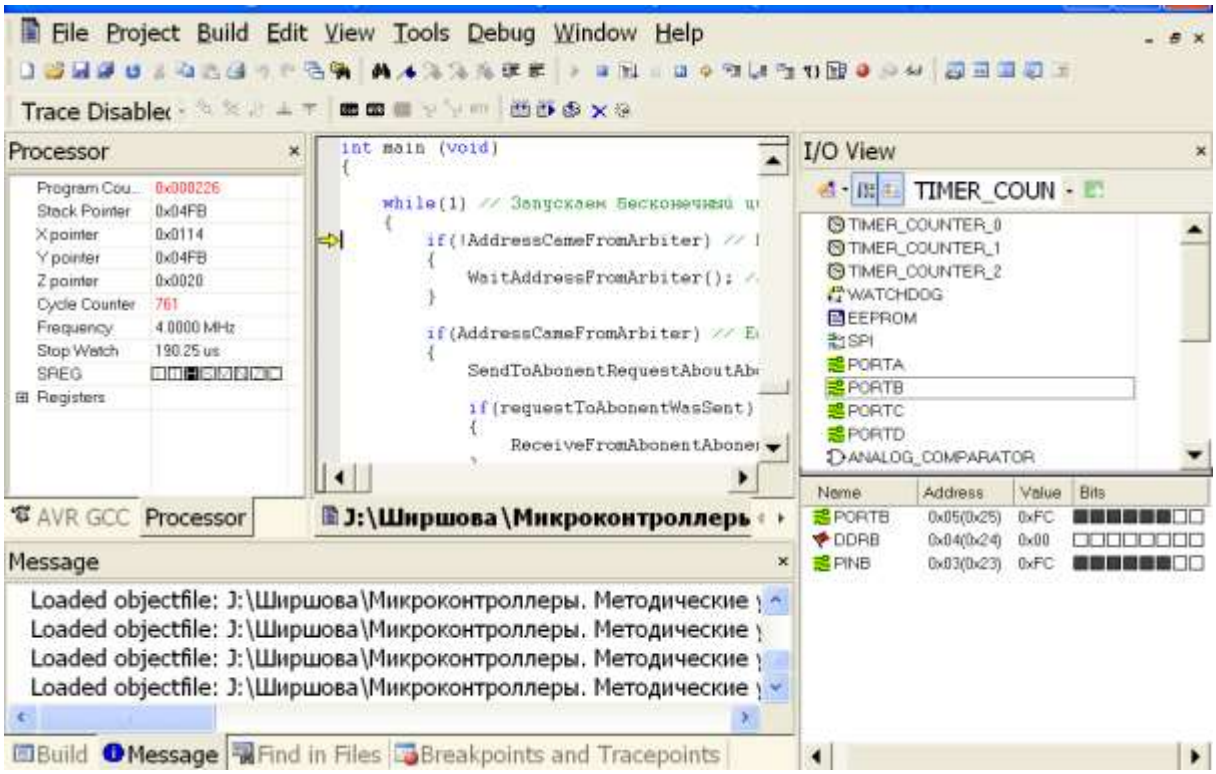


C2Out = true; // Запоминаем, что сигнал С 2 ушёл





if(!AddressCameFromArbiter) // Если адрес МК ещё не пришёл от Арбитра,


File Project Build Edit View Tools Debug Window Help
Trace Disable
Processor
Program Cou... 0x000226
Stack Pointer 0x04FB
X pointer 0x0114
Y pointer 0x04FB
Z pointer 0x0020
Cycle Counter 761
Frequency 4.0000 MHz
Stop Watch 190.25 us
SREG
Registers
int main (void)
{
while(1) // Запускаем бесконечный ц
{
if(!AddressCameFromArbiter) //
{
WaitAddressFromArbiter(); /
}
if(AddressCameFromArbiter) // E
{
SendToAbonentRequestAboutAb
if(requestToAbonentWasSent)
{
ReceiveFromAbonentAbone
I/O View
TIMER_COUN
TIMER_COUNTER_0
TIMER_COUNTER_1
TIMER_COUNTER_2
WATCHDOG
EEPROM
SPI
PORTA
PORTB
PORTC
PORTD
ANALOG_COMPARATOR
Name Address Value Bits
PORTB 0x05(0x25) 0xFC
DDRB 0x04(0x24) 0x00
PINB 0x03(0x23) 0xFC
AVR GCC Processor
J:\Ширшова\Микроконтроллеры
Message
Loaded objectfile: J:\Ширшова\Микроконтроллеры. Методические
Loaded objectfile: J:\Ширшова\Микроконтроллеры. Методические
Loaded objectfile: J:\Ширшова\Микроконтроллеры. Методические
Loaded objectfile: J:\Ширшова\Микроконтроллеры. Методические
Build Message Find in Files Breakpoints and Tracepoints