a2ps - A General Purpose PostScript Generating Utility

a2ps formats files for printing on a PostScript printer. It can "pretty print" plain text files, Spice files, just about any computer language, and even Matlab files. With each language, it highlights the different keywords. Its slogan is "Do The Right Thing", which means that though it is highly configurable, everything was setup so that a novice user can usually get the output they want without much effort.

The default format used is usually just what you were looking for. Typically, this means two pages per physical page with borders, and some sort of header. The basic syntax for a2ps from the command line is:

```
a2ps  options filename
```

To print the C file 'newkeyer.c', the default setting is probably just fine:

```
$ a2ps newkeyer.c
[newkeyer.c (C): 5 pages on 3 sheets]
[Total: 5 pages on 3 sheets] sent to the default printer
```

Printed by Traylor Roger

Sep 17, 14 15:43 **newkeyer.c** Page 1/5

```
// mykeyer.c
// R. Traylor
// 12.19.2010
// iambic keyer

#define F_CPU 8000000UL     //8Mhz clock
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Note: ATmega48 is supplied with 8 Mhz RC internal clock enabled with
//clkdiv programmed which sets internal clock to 1Mhz.

// I/O allocation
//ADC7  analog pin to read the speed value
//PC0   when low, dot paddle closed
//PC1   when low, dash paddle closed
//PC2   drives the driver xistor that closes the key connection
//PC3   mute2, relay output, when high, xmitter is connected to antenna
//PC4   mute1, mute output, when high, reciever is muted
//PC5   spare, unused

//PD0   encoder a pin
//PD1   encoder b pin
//PD3   pin 1, OC2B compare output for sidetone

//main keyer state machine states
#define IDLE     0  //waiting for a paddle closure
#define DIT      1  //making a dit
#define DAH      2  //making a dah
#define DIT_DLY  3  //intersymbol delay, one dot time
#define DAH_DLY  4  //intersymbol delay, one dot time
//
#define FALSE    0
#define TRUE     1
//

//define EEPROM storage areas
uint8_t EEMEM eeprom_wds_per_min = 20;    //this value only assigned during prog
ramming

volatile uint8_t  keyer_state   = IDLE;   //the keyer state
volatile uint8_t  timer_ena     = 0;      //the timer enable
volatile uint8_t  dit_pending   = FALSE;  //memory for dits
volatile uint8_t  dah_pending   = FALSE;  //memory for dahs
volatile uint16_t timeout       ;         //one dot interval
volatile uint16_t half_timeout  ;         //one half dot interval
volatile uint8_t  key           = FALSE;  //internal keyer output
volatile int16_t  ee_wait_cnt   = -1;     //countdown to save new setting to eep
rom

volatile uint8_t  wds_per_min = 20;       //words per minute (still need to inta
lize if in EEPROM?)

//output state machine states
#define IDLE   0  //
#define A      1  //
#define B      2  //
#define C      3  //
#define D      4  //
#define E      5  //
#define F      6  //
#define G      7  //
#define H      8  //
#define I      9  //

//output state machine variables
volatile uint8_t output_state  = IDLE;
```

Sep 17, 14 15:43 **newkeyer.c** Page 2/5

```
volatile uint8_t tx_dly        = 0x00;
volatile uint8_t mute1_timeout    = 4;  //500uS for audio mute to engage
volatile uint8_t relay_timeout    = 31; //4mS for relay to actuate
volatile uint8_t tx_decay         = 39; //5mS for xmit envelope to decay

//functions
void    tx_on()  {PORTC &= ~0b00000100;}           //asserts key output
void    tx_off() {PORTC |=  0b00000100;}           //deasserts key output
uint8_t dit_on() {return(bit_is_clear(PINC, 0));}  //returns non-zero if dit pad
dle on
uint8_t dah_on() {return(bit_is_clear(PINC, 1));}  //returns non-zero if dah pad
dle on

/*****************************  mute1  ***************************************/
void mute1(state){if(state==TRUE){PORTC |=  (1<<PC4);}
                  else           {PORTC &= ~(1<<PC4);}
}
/*****************************************************************************/

/*****************************  mute2  ***************************************/
void mute2(state){if(state==TRUE){PORTC |=  (1<<PC3);}
                  else           {PORTC &= ~(1<<PC3);}
}
/*****************************************************************************/

/*****************************************************************************/
//                              encoder_chk
//Checks for encoder movement.  Debounces encoder and returns a one exactly once
.
//for each falling edge detected on PD0.  Takes 12 cycles to determine movement.

//Note: Returns signed value.
//
int8_t encoder_chk() {

static uint16_t state = 0;  //holds shifted in bits from encoder

  state = (state << 1) | (! bit_is_clear(PIND, PD0)) | 0xE000;
  if(state == 0xF000){                        //if a falling edge is detected on A
output
    //get state of "B" to determine direction
    if (bit_is_set(PIND, PD1)){return 1;} //detected CW rotation
    else                      {return 0;} //detected CCW rotation
  }
  else{return -1;}                          //no movement detected
}//encoder_chk
/*****************************************************************************/

//*****************************************************************************/
//                  Interrupt service routine for timer 0
//worst case is about 18uS to run ISR

ISR(TIMER0_COMPA_vect){

static uint16_t timer;

  timer--;  //decrement clock

//keyer main state machine
  switch(keyer_state){ //see if user changed the minutes setting
    case (IDLE) :
      key = FALSE;
      if     (dit_on()){timer = timeout;   keyer_state = DIT;}
      else if(dah_on()){timer = timeout*3; keyer_state = DAH;}
      break;
    case (DIT) :
      key = TRUE;
      if(!timer){timer = timeout; keyer_state = DIT_DLY;}
      break;
```

Wednesday September 17, 2014 newkeyer.c 1/3

Figure 1: a2ps newkeyer.c

a2ps sent the file ‘newkeyer.c’ to the default printer, writing two columns of text on a single face of the sheet. Default a2ps uses the predefined option ‘-2’, meaning two virtual pages per physical page. If we wanted only one virtual page per physical page, the ”-1” option is specified:

```
$ a2ps -1 newkeyer.c
[newkeyer.c (C): 5 pages on 5 sheets]
[Total: 5 pages on 5 sheets] sent to the default printer
```

Printed by Traylor Roger

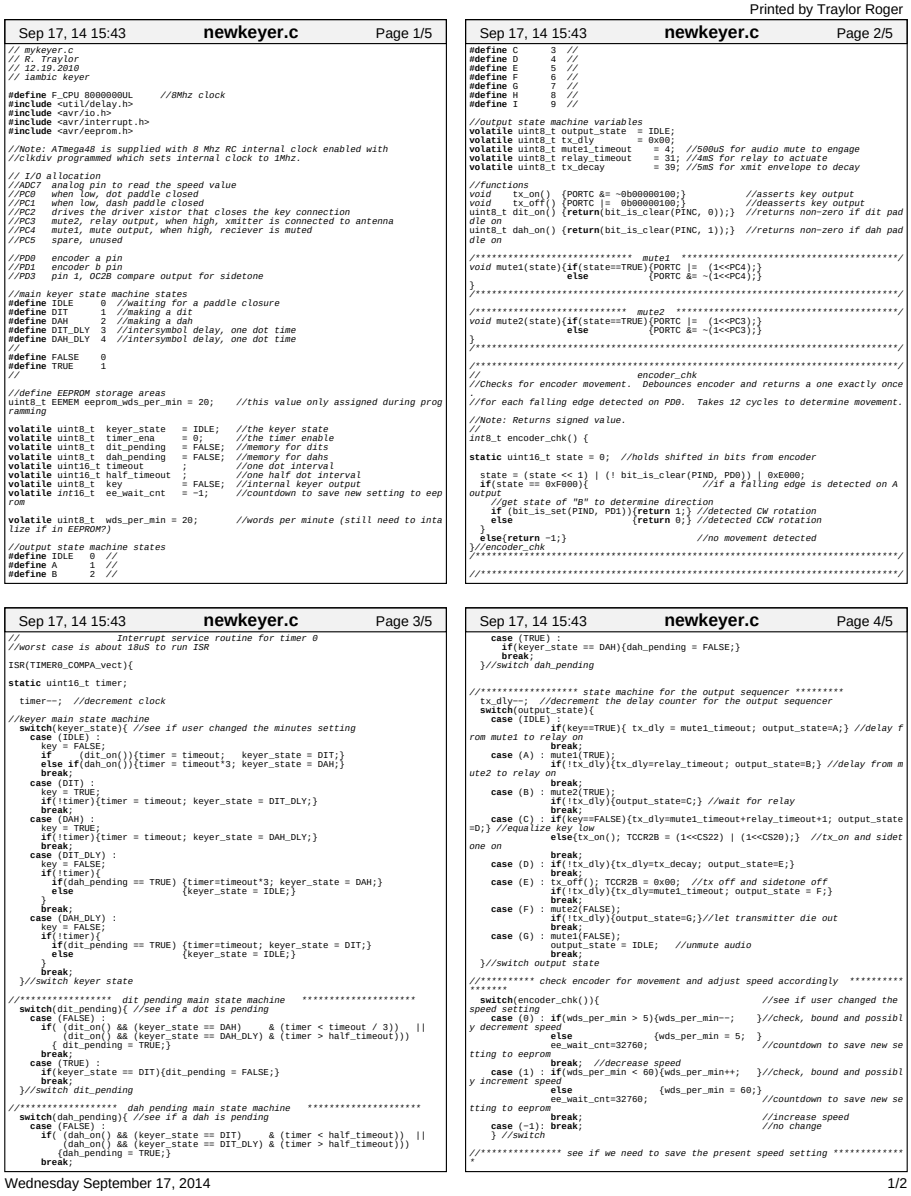
Sep 17, 14 15:43 **newkeyer.c** Page 1/5

```
// mykeyer.c
// R. Traylor
// 12.19.2010
// iambic keyer

#define F_CPU 8000000UL     //8Mhz clock
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Note: ATmega48 is supplied with 8 Mhz RC internal clock enabled with
//clkdiv programmed which sets internal clock to 1Mhz.

// I/O allocation
//ADC7  analog pin to read the speed value
//PC0   when low, dot paddle closed
//PC1   when low, dash paddle closed
//PC2   drives the driver xistor that closes the key connection
//PC3   mute2, relay output, when high, xmitter is connected to antenna
//PC4   mute1, mute output, when high, reciever is muted
//PC5   spare, unused

//PD0   encoder a pin
//PD1   encoder b pin
//PD3   pin 1, OC2B compare output for sidetone

//main keyer state machine states
#define IDLE     0  //waiting for a paddle closure
#define DIT      1  //making a dit
#define DAH      2  //making a dah
#define DIT_DLY  3  //intersymbol delay, one dot time
#define DAH_DLY  4  //intersymbol delay, one dot time
//
#define FALSE    0
#define TRUE     1
//

//define EEPROM storage areas
uint8_t EEMEM eeprom_wds_per_min = 20;    //this value only assigned during prog
ramming

volatile uint8_t  keyer_state   = IDLE;   //the keyer state
volatile uint8_t  timer_ena     = 0;      //the timer enable
volatile uint8_t  dit_pending   = FALSE;  //memory for dits
volatile uint8_t  dah_pending   = FALSE;  //memory for dahs
volatile uint16_t timeout       ;         //one dot interval
volatile uint16_t half_timeout  ;         //one half dot interval
volatile uint8_t  key           = FALSE;  //internal keyer output
volatile int16_t  ee_wait_cnt   = -1;     //countdown to save new setting to eep
rom

volatile uint8_t  wds_per_min = 20;       //words per minute (still need to inta
lize if in EEPROM?)

//output state machine states
#define IDLE   0  //
#define A      1  //
#define B      2  //
```

Wednesday September 17, 2014 1/5

Figure 2: `a2ps -1 newkeyer.c`

There are options for placing even more virtual pages on a physical page. However they begin to get very difficult to read as the font is scaled automatically. At more than four virtual pages, the text is just too small to read.



Printed by Traylor Roger

Sep 17, 14 15:43 **newkeyer.c** Page 1/5

```
// mykeyer.c
// R. Traylor
// 12.19.2010
// iambic keyer

#define F_CPU 8000000UL     //8Mhz clock
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Note: ATmega48 is supplied with 8 Mhz RC internal clock enabled with
//clkdiv programmed which sets internal clock to 1Mhz.

// I/O allocation
//ADC7  analog pin to read the speed value
//PC0   when low, dot paddle closed
//PC1   when low, dash paddle closed
//PC2   drives the driver xistor that closes the key connection
//PC3   mute2, relay output, when high, xmitter is connected to antenna
//PC4   mute1, mute output, when high, reciever is muted
//PC5   spare, unused

//PD0   encoder a pin
//PD1   encoder b pin
//PD3   pin 1, OC2B compare output for sidetone

//main keyer state machine states
#define IDLE    0  //waiting for a paddle closure
#define DIT     1  //making a dit
#define DAH     2  //making a dah
#define DIT_DLY 3  //intersymbol delay, one dot time
#define DAH_DLY 4  //intersymbol delay, one dot time
//
#define FALSE   0
#define TRUE    1
//

//define EEPROM storage areas
uint8_t EEMEM eeprom_wds_per_min = 20;    //this value only assigned during prog
ramming

volatile uint8_t  keyer_state   = IDLE;   //the keyer state
volatile uint8_t  timer_ena     = 0;      //the timer enable
volatile uint8_t  dit_pending   = FALSE;  //memory for dits
volatile uint8_t  dah_pending   = FALSE;  //memory for dahs
volatile uint16_t timeout       ;         //one dot interval
volatile uint16_t half_timeout  ;         //one half dot interval
volatile uint8_t  key           = FALSE;  //internal keyer output
volatile int16_t  ee_wait_cnt   = -1;     //countdown to save new setting to eep
rom

volatile uint8_t  wds_per_min = 20;       //words per minute (still need to inta
lize if in EEPROM?)

//output state machine states
#define IDLE   0  //
#define A      1  //
#define B      2  //
```

Sep 17, 14 15:43 **newkeyer.c** Page 2/5

```
#define C      3  //
#define D      4  //
#define E      5  //
#define F      6  //
#define G      7  //
#define H      8  //
#define I      9  //

//output state machine variables
volatile uint8_t output_state  = IDLE;
volatile uint8_t tx_dly        = 0x00;
volatile uint8_t mute1_timeout    = 4;  //500uS for audio mute to engage
volatile uint8_t relay_timeout    = 31; //4mS for relay to actuate
volatile uint8_t tx_decay         = 39; //5mS for xmit envelope to decay

//functions
void    tx_on()  {PORTC &= ~0b00000100;}          //asserts key output
void    tx_off() {PORTC |=  0b00000100;}          //deasserts key output
uint8_t dit_on() {return(bit_is_clear(PINC, 0));}  //returns non-zero if dit pad
dle on
uint8_t dah_on() {return(bit_is_clear(PINC, 1));}  //returns non-zero if dah pad
dle on

/*****************************  mute1  ***************************************/
void mute1(state){if(state==TRUE){PORTC |=  (1<<PC4);}
                  else           {PORTC &= ~(1<<PC4);}
}
/*****************************************************************************/

/*****************************  mute2  ***************************************/
void mute2(state){if(state==TRUE){PORTC |=  (1<<PC3);}
                  else           {PORTC &= ~(1<<PC3);}
}
/*****************************************************************************/

/*****************************************************************************/
//                          encoder_chk
//Checks for encoder movement.  Debounces encoder and returns a one exactly once
.
//for each falling edge detected on PD0.  Takes 12 cycles to determine movement.

//Note: Returns signed value.
//
int8_t encoder_chk() {

static uint16_t state = 0;  //holds shifted in bits from encoder

  state = (state << 1) | (! bit_is_clear(PIND, PD0)) | 0xE000;
  if(state == 0xF000){                    //if a falling edge is detected on A
output
    //get state of "B" to determine direction
    if (bit_is_set(PIND, PD1)){return 1;} //detected CW rotation
    else                      {return 0;} //detected CCW rotation
  }
  else{return -1;}                        //no movement detected
}//encoder_chk
/*****************************************************************************/

/*****************************************************************************/
```

Sep 17, 14 15:43 **newkeyer.c** Page 3/5

```
//                Interrupt service routine for timer 0
//worst case is about 18uS to run ISR

ISR(TIMER0_COMPA_vect){

static uint16_t timer;

  timer--;  //decrement clock

//keyer main state machine
  switch(keyer_state){ //see if user changed the minutes setting
    case (IDLE) :
      key = FALSE;
      if     (dit_on()){timer = timeout;   keyer_state = DIT;}
      else if(dah_on()){timer = timeout*3; keyer_state = DAH;}
      break;
    case (DIT) :
      key = TRUE;
      if(!timer){timer = timeout; keyer_state = DIT_DLY;}
      break;
    case (DAH) :
      key = TRUE;
      if(!timer){timer = timeout; keyer_state = DAH_DLY;}
      break;
    case (DIT_DLY) :
      key = FALSE;
      if(!timer){
        if(dah_pending == TRUE) {timer=timeout*3; keyer_state = DAH;}
        else                    {keyer_state = IDLE;}
      }
      break;
    case (DAH_DLY) :
      key = FALSE;
      if(!timer){
        if(dit_pending == TRUE) {timer=timeout; keyer_state = DIT;}
        else                    {keyer_state = IDLE;}
      }
      break;
  }//switch keyer state

//******************  dit pending main state machine   *********************
  switch(dit_pending){ //see if a dot is pending
    case (FALSE) :
      if( (dit_on() && (keyer_state == DAH)     & (timer < timeout / 3))  ||
          (dit_on() && (keyer_state == DAH_DLY) & (timer > half_timeout)))
        { dit_pending = TRUE;}
      break;
    case (TRUE) :
      if(keyer_state == DIT){dit_pending = FALSE;}
      break;
  }//switch dit_pending

//******************  dah pending main state machine   *********************
  switch(dah_pending){ //see if a dah is pending
    case (FALSE) :
      if( (dah_on() && (keyer_state == DIT)     & (timer < half_timeout))  ||
          (dah_on() && (keyer_state == DIT_DLY) & (timer > half_timeout)))
        {dah_pending = TRUE;}
      break;
```

Sep 17, 14 15:43 **newkeyer.c** Page 4/5

```
    case (TRUE) :
      if(keyer_state == DAH){dah_pending = FALSE;}
      break;
  }//switch dah_pending


//******************* state machine for the output sequencer *********
  tx_dly--;  //decrement the delay counter for the output sequencer
  switch(output_state){
    case (IDLE) :
                  if(key==TRUE){ tx_dly = mute1_timeout; output_state=A;} //delay f
rom mute1 to relay on
                  break;
    case (A) : mute1(TRUE);
                  if(!tx_dly){tx_dly=relay_timeout; output_state=B;} //delay from m
ute2 to relay on
                  break;
    case (B) : mute2(TRUE);
                  if(!tx_dly){output_state=C;} //wait for relay
                  break;
    case (C) : if(key==FALSE){tx_dly=mute1_timeout+relay_timeout+1; output_state
=D;} //equalize key low
                  else{tx_on(); TCCR2B = (1<<CS22) | (1<<CS20);}  //tx_on and sidet
one on
                  break;
    case (D) : if(!tx_dly){tx_dly=tx_decay; output_state=E;}
                  break;
    case (E) : tx_off(); TCCR2B = 0x00;  //tx off and sidetone off
                  if(!tx_dly){tx_dly=mute1_timeout; output_state = F;}
                  break;
    case (F) : mute2(FALSE);
                  if(!tx_dly){output_state=G;}//let transmitter die out
                  break;
    case (G) : mute1(FALSE);
                  output_state = IDLE;   //unmute audio
                  break;
  }//switch output state

//********** check encoder for movement and adjust speed accordingly  **********
*******
    switch(encoder_chk()){                            //see if user changed the
speed setting
      case (0) : if(wds_per_min > 5){wds_per_min--;    }//check, bound and possibl
y decrement speed
                 else                {wds_per_min = 5;  }
                 ee_wait_cnt=32760;                    //countdown to save new se
tting to eeprom
                 break;  //decrease speed
      case (1) : if(wds_per_min < 60){wds_per_min++;   }//check, bound and possibl
y increment speed
                 else                {wds_per_min = 60;}
                 ee_wait_cnt=32760;                    //countdown to save new se
tting to eeprom
                 break;                                //increase speed
      case (-1): break;                                //no change
    } //switch

//***************** see if we need to save the present speed setting *************
*
```

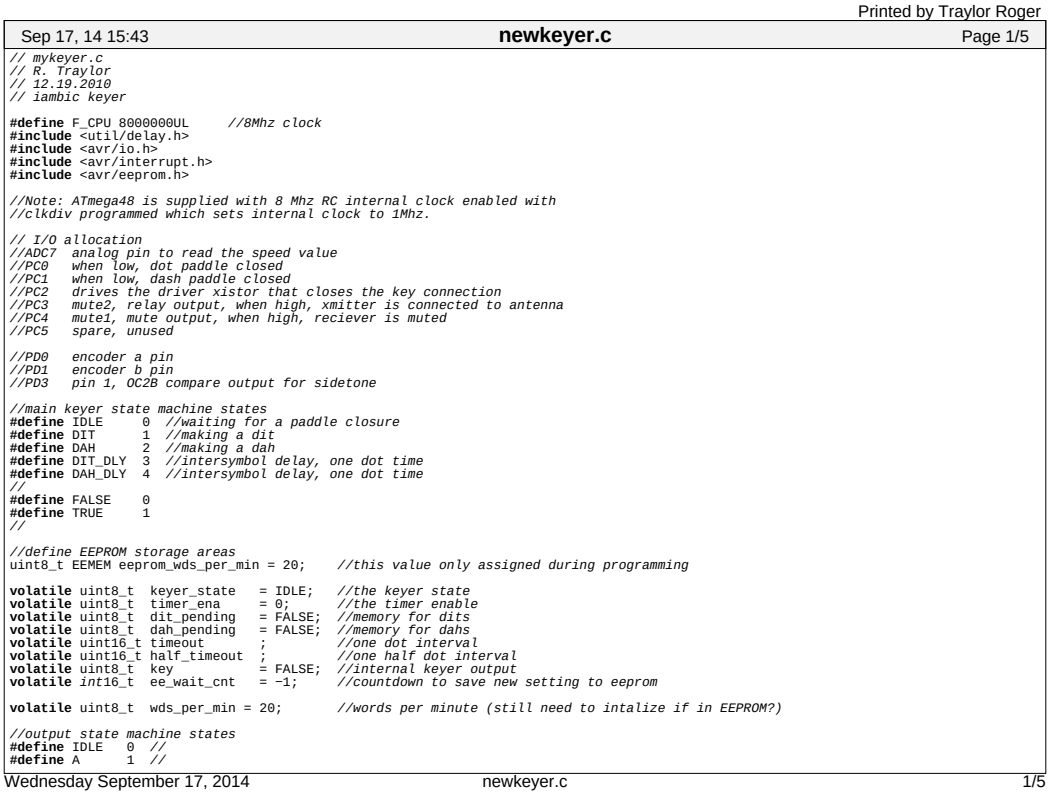

Figure 3: `a2ps -4 newkeyer.c`

a2ps can be more finely tuned by the user with more command line options. However, some command line options can interfere with each other, so they must be chosen intelligently. For example, we could choose a ”one-up” page but specify the font size to be 9 point get more text on a line.

```
// mykeyer.c
// R. Traylor
// 12.19.2010
// iambic keyer

#define F_CPU 8000000UL     //8Mhz clock
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Note: ATmega48 is supplied with 8 Mhz RC internal clock enabled with
//clkdiv programmed which sets internal clock to 1Mhz.

// I/O allocation
//ADC7  analog pin to read the speed value
//PC0   when low, dot paddle closed
//PC1   when low, dash paddle closed
//PC2   drives the driver xistor that closes the key connection
//PC3   mute2, relay output, when high, xmitter is connected to antenna
//PC4   mute1, mute output, when high, reciever is muted
//PC5   spare, unused

//PD0   encoder a pin
//PD1   encoder b pin
//PD3   pin 1, OC2B compare output for sidetone

//main keyer state machine states
#define IDLE     0  //waiting for a paddle closure
#define DIT      1  //making a dit
#define DAH      2  //making a dah
#define DIT_DLY  3  //intersymbol delay, one dot time
#define DAH_DLY  4  //intersymbol delay, one dot time
//
#define FALSE    0
#define TRUE     1
//

//define EEPROM storage areas
uint8_t EEMEM eeprom_wds_per_min = 20;    //this value only assigned during programming

volatile uint8_t  keyer_state   = IDLE;   //the keyer state
volatile uint8_t  timer_ena     = 0;      //the timer enable
volatile uint8_t  dit_pending   = FALSE;  //memory for dits
volatile uint8_t  dah_pending   = FALSE;  //memory for dahs
volatile uint16_t timeout       ;         //one dot interval
volatile uint16_t half_timeout  ;         //one half dot interval
volatile uint8_t  key           = FALSE;  //internal keyer output
volatile int16_t  ee_wait_cnt   = -1;     //countdown to save new setting to eeprom

volatile uint8_t  wds_per_min = 20;       //words per minute (still need to intalize if in EEPROM?)

//output state machine states
#define IDLE   0  //
#define A      1  //
#define B      2  //
#define C      3  //
#define D      4  //
#define E      5  //
#define F      6  //
#define G      7  //
#define H      8  //
#define I      9  //

//output state machine variables
volatile uint8_t output_state  = IDLE;
volatile uint8_t tx_dly        = 0x00;
volatile uint8_t mute1_timeout    = 4;  //500uS for audio mute to engage
volatile uint8_t relay_timeout    = 31; //4mS for relay to actuate
volatile uint8_t tx_decay         = 39; //5mS for xmit envelope to decay

//functions
void    tx_on()  {PORTC &= ~0b00000100;}          //asserts key output
void    tx_off() {PORTC |=  0b00000100;}          //deasserts key output
uint8_t dit_on() {return(bit_is_clear(PINC, 0));}  //returns non-zero if dit paddle on
uint8_t dah_on() {return(bit_is_clear(PINC, 1));}  //returns non-zero if dah paddle on
```


Figure 4: `a2ps -P printer -1 --font-size=9 newkeyer.c`

If your code tends to have even longer lines, you may want to rotate the page to landscape mode and allow printing of up to 132 characters per line. Here we would not specify a font size as a2ps will choose the font size to provide space for 132 characters per line.



Printed by Traylor Roger

Sep 17, 14 15:43 **newkeyer.c** Page 1/5

```
// mykeyer.c
// R. Traylor
// 12.19.2010
// iambic keyer

#define F_CPU 8000000UL     //8Mhz clock
#include <util/delay.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

//Note: ATmega48 is supplied with 8 Mhz RC internal clock enabled with
//clkdiv programmed which sets internal clock to 1Mhz.

// I/O allocation
//ADC7  analog pin to read the speed value
//PC0   when low, dot paddle closed
//PC1   when low, dash paddle closed
//PC2   drives the driver xistor that closes the key connection
//PC3   mute2, relay output, when high, xmitter is connected to antenna
//PC4   mute1, mute output, when high, reciever is muted
//PC5   spare, unused

//PD0   encoder a pin
//PD1   encoder b pin
//PD3   pin 1, OC2B compare output for sidetone

//main keyer state machine states
#define IDLE     0  //waiting for a paddle closure
#define DIT      1  //making a dit
#define DAH      2  //making a dah
#define DIT_DLY  3  //intersymbol delay, one dot time
#define DAH_DLY  4  //intersymbol delay, one dot time
//
#define FALSE    0
#define TRUE     1
//

//define EEPROM storage areas
uint8_t EEMEM eeprom_wds_per_min = 20;    //this value only assigned during programming

volatile uint8_t  keyer_state   = IDLE;   //the keyer state
volatile uint8_t  timer_ena     = 0;      //the timer enable
volatile uint8_t  dit_pending   = FALSE;  //memory for dits
volatile uint8_t  dah_pending   = FALSE;  //memory for dahs
volatile uint16_t timeout       ;         //one dot interval
volatile uint16_t half_timeout  ;         //one half dot interval
volatile uint8_t  key           = FALSE;  //internal keyer output
volatile int16_t  ee_wait_cnt   = -1;     //countdown to save new setting to eeprom

volatile uint8_t  wds_per_min = 20;       //words per minute (still need to intalize if in EEPROM?)

//output state machine states
#define IDLE   0  //
#define A      1  //
```

Wednesday September 17, 2014 newkeyer.c 1/5

Figure 5: `a2ps -P printer --landscape --columns=1 --chars-per-line=132 newkeyer.c`

There are many options that can be used. To see them all, at the prompt, type `man a2ps`, or `a2ps --help`. Here are a few peculiar examples that may be useful.

```
a2ps -P void file.c           //send output to trash, but see how many pages it makes
a2ps -P display file.c        //send to ghostview to see how it looks before printing
a2ps -P -o outputfile file.c  //send postscript output to a file called "outputfile"
a2ps -P pdf file.c            //print a2ps formatted pdf output to file.pdf
```

Finally, a short list of the most useful options.

```
Help:
-h            same as --help

Printing:
-o myfile     OR  --output=myfile       //write output to a named file
-P myprinter  OR  --printer=myprinter   //send a2ps output to named printer

Formatting:
-r            OR  --landscape          //print in landscape mode
-R            OR  --portrait           //print in portrait mode
-f NUM        OR  --font-size=NUM      //set font size
-L NUM        OR  --lines-per-page=NUM //scale font to print NUM lines per virtual page
-c NUM        OR  --columns=NUM        //number of virtual pages per physical page
-B            OR  --no-header          //no header to be printed
-b            OR  --header             //header will be printed
```