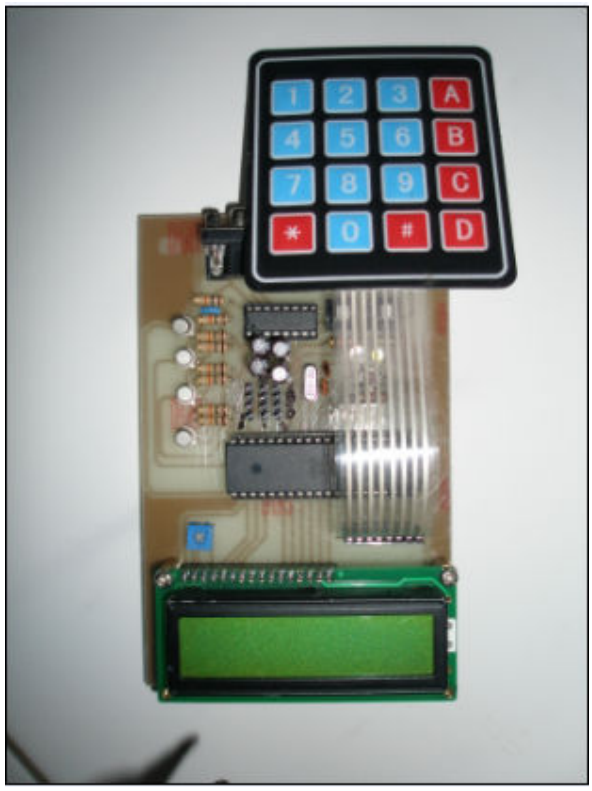
LAMPIRAN A
FOTO RANGKAIAN PENGUKURAN HASIL PRODUKSI KAIN DAN MESIN *FINISHING* TEKSTIL

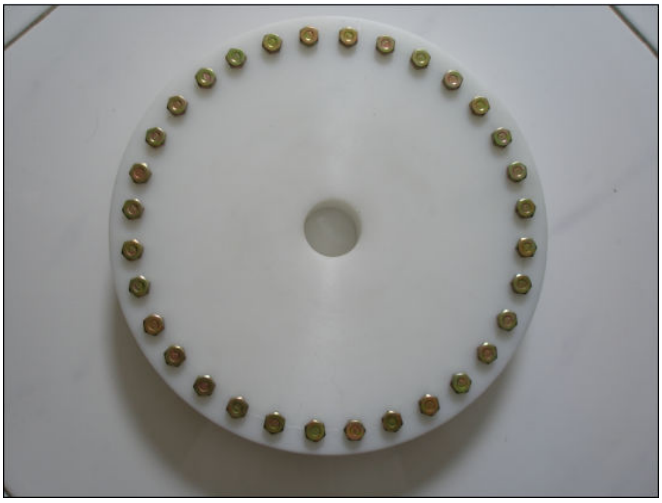
MESIN SUEDING

MESIN CALENDER

RANGKAIAN PENGUKURAN PANJANG PRODUKSI KAIN



PIRINGAN BAUT



LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO ATMEGA16

Make File

```
# Hey Emacs, this is a -*- makefile -*-
#----------------------------------------------------------------------------
# WinAVR Makefile Template written by Eric B. Weddington, Jörg Wunsch, et al.
#
# Released to the Public Domain
#
# Additional material for this makefile was written by:
# Peter Fleury
# Tim Henigan
# Colin O'Flynn
# Reiner Patommel
# Markus Pfaff
# Sander Pool
# Frederik Rouleau
# Carlos Lamas
#
#----------------------------------------------------------------------------
# On command line:
#
# make all = Make software.
#
# make clean = Clean out built project files.
#
# make coff = Convert ELF to AVR COFF.
#
# make extcoff = Convert ELF to AVR Extended COFF.
#
# make program = Download the hex file to the device, using avrdude.
#                Please customize the avrdude settings below first!
#
# make debug = Start either simulavr or avarice as specified for debugging,
#              with avr-gdb or avr-insight as the front end for debugging.
#
# make filename.s = Just compile filename.c into the assembler code only.
#
# make filename.i = Create a preprocessed source file for use in submitting
#                   bug reports to the GCC project.
#
# To rebuild project do "make clean" then "make all".
#----------------------------------------------------------------------------


# MCU name
MCU = atmega16


# Processor frequency.
#     This will define a symbol, F_CPU, in all source code files equal to the
#     processor frequency. You can then use this symbol in your source code to
#     calculate timings. Do NOT tack on a 'UL' at the end, this will be done
#     automatically to create a 32-bit value in your source code.
#     Typical values are:
#         F_CPU =  1000000
#         F_CPU =  1843200
#         F_CPU =  2000000
#         F_CPU =  3686400
#         F_CPU =  4000000
#F_CPU =  7372800
#         F_CPU =  8000000
F_CPU = 11059200
#         F_CPU = 14745600
#F_CPU = 16000000
#         F_CPU = 18432000
#         F_CPU = 20000000
#         F_CPU = 8000000
```

```
# Output format. (can be srec, ihex, binary)
FORMAT = ihex


# Target file name (without extension).
TARGET = scheduler


# Object files directory
#     To put object files in current directory, use a dot (.), do NOT make
#     this an empty or blank macro!
OBJDIR = .


# List C source files here. (C dependencies are automatically generated.)
SRC = $(TARGET).c lcd.c


# List C++ source files here. (C dependencies are automatically generated.)
CPPSRC =


# List Assembler source files here.
#     Make them always end in a capital .S.  Files ending in a lowercase .s
#     will not be considered source files but generated files (assembler
#     output from the compiler), and will be deleted upon "make clean"!
#     Even though the DOS/Win* filesystem matches both .s and .S the same,
#     it will preserve the spelling of the filenames, and gcc itself does
#     care about how the name is spelled on its command-line.
ASRC =


# Optimization level, can be [0, 1, 2, 3, s].
#     0 = turn off optimization. s = optimize for size.
#     (Note: 3 is not always the best optimization level. See avr-libc FAQ.)
OPT = s


# Debugging format.
#     Native formats for AVR-GCC's -g are dwarf-2 [default] or stabs.
#     AVR Studio 4.10 requires dwarf-2.
#     AVR [Extended] COFF format requires stabs, plus an avr-objcopy run.
DEBUG = dwarf-2


# List any extra directories to look for include files here.
#     Each directory must be seperated by a space.
#     Use forward slashes for directory separators.
#     For a directory that has spaces, enclose it in quotes.
EXTRAINCDIRS =


# Compiler flag to set the C Standard level.
#     c89   = "ANSI" C
#     gnu89 = c89 plus GCC extensions
#     c99   = ISO C99 standard (not yet fully implemented)
#     gnu99 = c99 plus GCC extensions
CSTANDARD = -std=gnu99


# Place -D or -U options here for C sources
CDEFS = -DF_CPU=$(F_CPU)UL


# Place -D or -U options here for C++ sources
```

```
CPPDEFS = -DF_CPU=$(F_CPU)UL
#CPPDEFS += -D__STDC_LIMIT_MACROS
#CPPDEFS += -D__STDC_CONSTANT_MACROS



#---------------- Compiler Options C ----------------
#  -g*:         generate debugging information
#  -O*:         optimization level
#  -f...:       tuning, see GCC manual and avr-libc documentation
#  -Wall...:    warning level
#  -Wa,...:     tell GCC to pass this to the assembler.
#    -adhlns...: create assembler listing
CFLAGS = -g$(DEBUG)
CFLAGS += $(CDEFS)
CFLAGS += -O$(OPT)
CFLAGS += -funsigned-char
CFLAGS += -funsigned-bitfields
CFLAGS += -fpack-struct
CFLAGS += -fshort-enums
CFLAGS += -Wall
CFLAGS += -Wstrict-prototypes
#CFLAGS += -mshort-calls
#CFLAGS += -fno-unit-at-a-time
#CFLAGS += -Wundef
#CFLAGS += -Wunreachable-code
#CFLAGS += -Wsign-compare
CFLAGS += -Wa,-adhlns=$(<:%.c=$(OBJDIR)/%.lst)
CFLAGS += $(patsubst %,-I%,$(EXTRAINCDIRS))
CFLAGS += $(CSTANDARD)


#---------------- Compiler Options C++ ----------------
#  -g*:         generate debugging information
#  -O*:         optimization level
#  -f...:       tuning, see GCC manual and avr-libc documentation
#  -Wall...:    warning level
#  -Wa,...:     tell GCC to pass this to the assembler.
#    -adhlns...: create assembler listing
CPPFLAGS = -g$(DEBUG)
CPPFLAGS += $(CPPDEFS)
CPPFLAGS += -O$(OPT)
CPPFLAGS += -funsigned-char
CPPFLAGS += -funsigned-bitfields
CPPFLAGS += -fpack-struct
CPPFLAGS += -fshort-enums
CPPFLAGS += -fno-exceptions
CPPFLAGS += -Wall
CFLAGS += -Wundef
#CPPFLAGS += -mshort-calls
#CPPFLAGS += -fno-unit-at-a-time
#CPPFLAGS += -Wstrict-prototypes
#CPPFLAGS += -Wunreachable-code
#CPPFLAGS += -Wsign-compare
CPPFLAGS += -Wa,-adhlns=$(<:%.cpp=$(OBJDIR)/%.lst)
CPPFLAGS += $(patsubst %,-I%,$(EXTRAINCDIRS))
#CPPFLAGS += $(CSTANDARD)


#---------------- Assembler Options ----------------
#  -Wa,...:   tell GCC to pass this to the assembler.
#  -ahlms:    create listing
#  -gstabs:   have the assembler create line number information; note that
#             for use in COFF files, additional information about filenames
#             and function names needs to be present in the assembler source
#             files -- see avr-libc docs [FIXME: not yet described there]
#  -listing-cont-lines: Sets the maximum number of continuation lines of hex
```

```
#       dump that will be displayed for a given single line of source input.
ASFLAGS = -Wa,-adhlns=$(<:.S=$(OBJDIR)/%.lst),-gstabs,--listing-cont-lines=100


#---------------- Library Options ----------------
# Minimalistic printf version
PRINTF_LIB_MIN = -Wl,-u,vfprintf -lprintf_min

# Floating point printf version (requires MATH_LIB = -lm below)
PRINTF_LIB_FLOAT = -Wl,-u,vfprintf -lprintf_flt

# If this is left blank, then it will use the Standard printf version.
PRINTF_LIB = $(PRINTF_LIB_FLOAT)
#PRINTF_LIB = $(PRINTF_LIB_MIN)
#PRINTF_LIB = $(PRINTF_LIB_FLOAT)


# Minimalistic scanf version
SCANF_LIB_MIN = -Wl,-u,vfscanf -lscanf_min

# Floating point + %[ scanf version (requires MATH_LIB = -lm below)
SCANF_LIB_FLOAT = -Wl,-u,vfscanf -lscanf_flt

# If this is left blank, then it will use the Standard scanf version.
SCANF_LIB =
#SCANF_LIB = $(SCANF_LIB_MIN)
#SCANF_LIB = $(SCANF_LIB_FLOAT)


MATH_LIB = -lm


# List any extra directories to look for libraries here.
#     Each directory must be seperated by a space.
#     Use forward slashes for directory separators.
#     For a directory that has spaces, enclose it in quotes.
EXTRALIBDIRS =



#---------------- External Memory Options ----------------

# 64 KB of external RAM, starting after internal RAM (ATmega128!),
# used for variables (.data/.bss) and heap (malloc()).
#EXTMEMOPTS = -Wl,-Tdata=0x801100,--defsym=__heap_end=0x80ffff

# 64 KB of external RAM, starting after internal RAM (ATmega128!),
# only used for heap (malloc()).
#EXTMEMOPTS = -Wl,--section-start,.data=0x801100,--defsym=__heap_end=0x80ffff

EXTMEMOPTS =



#---------------- Linker Options ----------------
#  -Wl,...:     tell GCC to pass this to linker.
#    -Map:      create map file
#    --cref:    add cross reference to  map file
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref
LDFLAGS += $(EXTMEMOPTS)
LDFLAGS += $(patsubst %,-L%,$(EXTRALIBDIRS))
LDFLAGS += $(PRINTF_LIB) $(SCANF_LIB) $(MATH_LIB)
#LDFLAGS += -T linker_script.x



#---------------- Programming Options (avrdude) ----------------
```

```
# Programming hardware: alf avr910 avrisp bascom bsd
# dt006 pavr picoweb pony-stk200 sp12 stk200 stk500
#
# Type: avrdude -c ?
# to get a full listing.
#
AVRDUDE_PROGRAMMER = stk200

# com1 = serial port. Use lpt1 to connect to parallel port.
AVRDUDE_PORT = lpt1    # programmer connected to serial device

AVRDUDE_WRITE_FLASH = -U flash:w:$(TARGET).hex
#AVRDUDE_WRITE_EEPROM = -U eeprom:w:$(TARGET).eep


# Uncomment the following if you want avrdude's erase cycle counter.
# Note that this counter needs to be initialized first using -Yn,
# see avrdude manual.
#AVRDUDE_ERASE_COUNTER = -y

# Uncomment the following if you do /not/ wish a verification to be
# performed after programming the device.
#AVRDUDE_NO_VERIFY = -V

# Increase verbosity level.  Please use this when submitting bug
# reports about avrdude. See <http://savannah.nongnu.org/projects/avrdude>
# to submit bug reports.
#AVRDUDE_VERBOSE = -v -v

AVRDUDE_FLAGS = -p $(MCU) -P $(AVRDUDE_PORT) -c $(AVRDUDE_PROGRAMMER)
AVRDUDE_FLAGS += $(AVRDUDE_NO_VERIFY)
AVRDUDE_FLAGS += $(AVRDUDE_VERBOSE)
AVRDUDE_FLAGS += $(AVRDUDE_ERASE_COUNTER)



#---------------- Debugging Options ----------------

# For simulavr only - target MCU frequency.
DEBUG_MFREQ = $(F_CPU)

# Set the DEBUG_UI to either gdb or insight.
# DEBUG_UI = gdb
DEBUG_UI = insight

# Set the debugging back-end to either avarice, simulavr.
DEBUG_BACKEND = avarice
#DEBUG_BACKEND = simulavr

# GDB Init Filename.
GDBINIT_FILE = __avr_gdbinit

# When using avarice settings for the JTAG
JTAG_DEV = /dev/com1

# Debugging port used to communicate between GDB / avarice / simulavr.
DEBUG_PORT = 4242

# Debugging host used to communicate between GDB / avarice / simulavr, normally
#     just set to localhost unless doing some sort of crazy debugging when
#     avarice is running on a different computer.
DEBUG_HOST = localhost



#============================================================================
```

```
# Define programs and commands.
SHELL = sh
CC = avr-gcc
OBJCOPY = avr-objcopy
OBJDUMP = avr-objdump
SIZE = avr-size
AR = avr-ar rcs
NM = avr-nm
AVRDUDE = avrdude
REMOVE = rm -f
REMOVEDIR = rm -rf
COPY = cp
WINSHELL = cmd


# Define Messages
# English
MSG_ERRORS_NONE = Errors: none
MSG_BEGIN = -------- begin --------
MSG_END = --------  end  --------
MSG_SIZE_BEFORE = Size before:
MSG_SIZE_AFTER = Size after:
MSG_COFF = Converting to AVR COFF:
MSG_EXTENDED_COFF = Converting to AVR Extended COFF:
MSG_FLASH = Creating load file for Flash:
MSG_EEPROM = Creating load file for EEPROM:
MSG_EXTENDED_LISTING = Creating Extended Listing:
MSG_SYMBOL_TABLE = Creating Symbol Table:
MSG_LINKING = Linking:
MSG_COMPILING = Compiling C:
MSG_COMPILING_CPP = Compiling C++:
MSG_ASSEMBLING = Assembling:
MSG_CLEANING = Cleaning project:
MSG_CREATING_LIBRARY = Creating library:




# Define all object files.
OBJ = $(SRC:%.c=$(OBJDIR)/%.o) $(CPPSRC:%.cpp=$(OBJDIR)/%.o) $(ASRC:%.S=$(OBJDIR)/%.o)

# Define all listing files.
LST = $(SRC:%.c=$(OBJDIR)/%.lst) $(CPPSRC:%.cpp=$(OBJDIR)/%.lst) $(ASRC:%.S=$(OBJDIR)/%.lst)


# Compiler flags to generate dependency files.
GENDEPFLAGS = -MMD -MP -MF .dep/$(@F).d


# Combine all necessary flags and optional flags.
# Add target processor to flags.
ALL_CFLAGS = -mmcu=$(MCU) -I. $(CFLAGS) $(GENDEPFLAGS)
ALL_CPPFLAGS = -mmcu=$(MCU) -I. -x c++ $(CPPFLAGS) $(GENDEPFLAGS)
ALL_ASFLAGS = -mmcu=$(MCU) -I. -x assembler-with-cpp $(ASFLAGS)




# Default target.
all: begin gccversion sizebefore build sizeafter end

# Change the build target to build a HEX file or a library.
build: elf hex eep lss sym
#build: lib
```

```
elf: $(TARGET).elf
hex: $(TARGET).hex
eep: $(TARGET).eep
lss: $(TARGET).lss
sym: $(TARGET).sym
LIBNAME=lib$(TARGET).a
lib: $(LIBNAME)



# Eye candy.
# AVR Studio 3.x does not check make's exit code but relies on
# the following magic strings to be generated by the compile job.
begin:
	@echo
	@echo $(MSG_BEGIN)

end:
	@echo $(MSG_END)
	@echo


# Display size of file.
HEXSIZE = $(SIZE) --target=$(FORMAT) $(TARGET).hex
ELFSIZE = $(SIZE) --format=avr $(TARGET).elf

sizebefore:
	@if test -f $(TARGET).elf; then echo; echo $(MSG_SIZE_BEFORE); $(ELFSIZE); \
	2>/dev/null; echo; fi

sizeafter:
	@if test -f $(TARGET).elf; then echo; echo $(MSG_SIZE_AFTER); $(ELFSIZE); \
	2>/dev/null; echo; fi



# Display compiler version information.
gccversion :
	@$(CC) --version



# Program the device.
program: $(TARGET).hex $(TARGET).eep
	$(AVRDUDE) $(AVRDUDE_FLAGS) $(AVRDUDE_WRITE_FLASH) $(AVRDUDE_WRITE_EEPROM)


# Generate avr-gdb config/init file which does the following:
#     define the reset signal, load the target file, connect to target, and set
#     a breakpoint at main().
gdb-config:
	@$(REMOVE) $(GDBINIT_FILE)
	@echo define reset >> $(GDBINIT_FILE)
	@echo SIGNAL SIGHUP >> $(GDBINIT_FILE)
	@echo end >> $(GDBINIT_FILE)
	@echo file $(TARGET).elf >> $(GDBINIT_FILE)
	@echo target remote $(DEBUG_HOST):$(DEBUG_PORT)  >> $(GDBINIT_FILE)
ifeq ($(DEBUG_BACKEND),simulavr)
	@echo load  >> $(GDBINIT_FILE)
endif
	@echo break main >> $(GDBINIT_FILE)

debug: gdb-config $(TARGET).elf
ifeq ($(DEBUG_BACKEND), avarice)
	@echo Starting AVaRICE - Press enter when "waiting to connect" message displays.
```

```
	@$(WINSHELL) /c start avarice --jtag $(JTAG_DEV) --erase --program --file \
	$(TARGET).elf $(DEBUG_HOST):$(DEBUG_PORT)
	@$(WINSHELL) /c pause

else
	@$(WINSHELL) /c start simulavr --gdbserver --device $(MCU) --clock-freq \
	$(DEBUG_MFREQ) --port $(DEBUG_PORT)
endif
	@$(WINSHELL) /c start avr-$(DEBUG_UI) --command=$(GDBINIT_FILE)



# Convert ELF to COFF for use in debugging / simulating in AVR Studio or VMLAB.
COFFCONVERT = $(OBJCOPY) --debugging
COFFCONVERT += --change-section-address .data-0x800000
COFFCONVERT += --change-section-address .bss-0x800000
COFFCONVERT += --change-section-address .noinit-0x800000
COFFCONVERT += --change-section-address .eeprom-0x810000



coff: $(TARGET).elf
	@echo
	@echo $(MSG_COFF) $(TARGET).cof
	$(COFFCONVERT) -O coff-avr $< $(TARGET).cof


extcoff: $(TARGET).elf
	@echo
	@echo $(MSG_EXTENDED_COFF) $(TARGET).cof
	$(COFFCONVERT) -O coff-ext-avr $< $(TARGET).cof



# Create final output files (.hex, .eep) from ELF output file.
%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) -R .eeprom $< $@

%.eep: %.elf
	@echo
	@echo $(MSG_EEPROM) $@
	-$(OBJCOPY) -j .eeprom --set-section-flags=.eeprom="alloc,load" \
	--change-section-lma .eeprom=0 --no-change-warnings -O $(FORMAT) $< $@ || exit 0

# Create extended listing file from ELF output file.
%.lss: %.elf
	@echo
	@echo $(MSG_EXTENDED_LISTING) $@
	$(OBJDUMP) -h -S $< > $@

# Create a symbol table from ELF output file.
%.sym: %.elf
	@echo
	@echo $(MSG_SYMBOL_TABLE) $@
	$(NM) -n $< > $@



# Create library from object files.
.SECONDARY : $(TARGET).a
.PRECIOUS : $(OBJ)
%.a: $(OBJ)
	@echo
	@echo $(MSG_CREATING_LIBRARY) $@
```

```
	$(AR) $@ $(OBJ)


# Link: create ELF output file from object files.
.SECONDARY : $(TARGET).elf
.PRECIOUS : $(OBJ)
%.elf: $(OBJ)
	@echo
	@echo $(MSG_LINKING) $@
	$(CC) $(ALL_CFLAGS) $^ --output $@ $(LDFLAGS)


# Compile: create object files from C source files.
$(OBJDIR)/%.o : %.c
	@echo
	@echo $(MSG_COMPILING) $<
	$(CC) -c $(ALL_CFLAGS) $< -o $@


# Compile: create object files from C++ source files.
$(OBJDIR)/%.o : %.cpp
	@echo
	@echo $(MSG_COMPILING_CPP) $<
	$(CC) -c $(ALL_CPPFLAGS) $< -o $@


# Compile: create assembler files from C source files.
%.s : %.c
	$(CC) -S $(ALL_CFLAGS) $< -o $@


# Compile: create assembler files from C++ source files.
%.s : %.cpp
	$(CC) -S $(ALL_CPPFLAGS) $< -o $@


# Assemble: create object files from assembler source files.
$(OBJDIR)/%.o : %.S
	@echo
	@echo $(MSG_ASSEMBLING) $<
	$(CC) -c $(ALL_ASFLAGS) $< -o $@


# Create preprocessed source for use in sending a bug report.
%.i : %.c
	$(CC) -E -mmcu=$(MCU) -I. $(CFLAGS) $< -o $@


# Target: clean project.
clean: begin clean_list end

clean_list :
	@echo
	@echo $(MSG_CLEANING)
	$(REMOVE) $(TARGET).hex
	$(REMOVE) $(TARGET).eep
	$(REMOVE) $(TARGET).cof
	$(REMOVE) $(TARGET).elf
	$(REMOVE) $(TARGET).map
	$(REMOVE) $(TARGET).sym
	$(REMOVE) $(TARGET).lss
	$(REMOVE) $(SRC:%.c=$(OBJDIR)/%.o)
	$(REMOVE) $(SRC:%.c=$(OBJDIR)/%.lst)
	$(REMOVE) $(SRC:.c=.s)
	$(REMOVE) $(SRC:.c=.d)
	$(REMOVE) $(SRC:.c=.i)
	$(REMOVEDIR) .dep
```

```
# Create object files directory
$(shell mkdir $(OBJDIR) 2>/dev/null)


# Include the dependency files.
-include $(shell mkdir .dep 2>/dev/null) $(wildcard .dep/*)


# Listing of phony targets.
.PHONY : all begin finish end sizebefore sizeafter gccversion \
build elf hex eep lss sym coff extcoff \
clean clean_list program debug gdb-config
```

PROGRAM UTAMA

```
/// Includes //////////
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <string.h>
#include <stdio.h>
#include "lcd.h"

#define BAUD 9600UL
#define MYUBRR ((F_CPU/(BAUD<<4))-1)

#define MENU 0
#define SEND_PC 1
#define RUN 2

#define MAX_SAVE 50

#define  _BV(bit)   (1 << (bit))
#define  inb(sfr)   _SFR_BYTE(sfr)
#define  inw(sfr)   _SFR_WORD(sfr)
#define  outb(sfr, val)   (_SFR_BYTE(sfr) = (val))
#define  outw(sfr, val)   (_SFR_WORD(sfr) = (val))
#define  cbi(sfr, bit)   (_SFR_BYTE(sfr) &= ~_BV(bit))
#define  sbi(sfr, bit)   (_SFR_BYTE(sfr) |= _BV(bit))
#define  toggle(sfr, bit)   (_SFR_BYTE(sfr) ^= _BV(bit))
#define  loop_until_bit_is_set(sfr, bit)   do { } while (bit_is_clear(sfr, bit))
#define  loop_until_bit_is_clear(sfr, bit)   do { } while (bit_is_set(sfr, bit))
#define  outp(val, sfr)   outb(sfr, val)
#define  inp(sfr)   inb(sfr)
#define  BV(bit)   _BV(bit)

/// Data Types /////////
typedef unsigned char u8;
typedef unsigned int u16;

#define key1 7
#define key2 6
#define key3 5
#define key4 4

#define sensor1 7
#define sensor2 6
#define sensor3 5
#define sensor4 4

#define MAXTIME 4

// time in minute
u8 time[5] = {1,2,3,4,5};
```

```
char buff[200];
u8 count_time;
u8 state;
u8 menuindex;
u8 savetime;
u8 rundisplay;

u8 k1_old, k2_old, k3_old, k4_old;
u8 k1, k2, k3, k4;

u8 s1_old, s2_old, s3_old, s4_old;
u8 s1, s2, s3, s4;

u16 count1, count2, count3, count4;
u16 count1_old, count2_old, count3_old, count4_old;

u8 save_count;
u16 address;
u16 data1, data2, data3, data4;
u8 num_data;
u8 send_count;

typedef struct task
{
	// pointer to a function
	void (*pfunc) (void);
	// delay before the first call
	u16 delay;
	// interval between subsequent runs
	u16 period;
	// flag indicating time to run
	u8 run;
}task;

// 10msec period 256-108
// 11.0592MHz and /1024 prescaler
#define StartFrom	148
// maximum number of tasks
#define MAXnTASKS	15


volatile task TaskArray[MAXnTASKS];

char lcd_line1[25];
char lcd_line2[25];
u8 lcd_state = 0;

/// Prototypes ////////
void USART_Init(void);
void TransmitByte(u8 data);
void TransmitString(char *);

u8 ReceiveByte(void);

void InitScheduler (void);
void UpdateScheduler(void);
void DeleteTask (u8 index);
void AddTask (void (*taskfunc)(void), u16 taskdelay, u16 taskperiod);
void DispatchTask (void);
void initKeypad(void);
void initSensor(void);
void initVar(void);

void eeprom_write_16bit(uint16_t address, uint16_t value);
uint16_t eeprom_read_16bit(uint16_t address);

void update_lcd(void);
```

```
void lcd_display(void);
void handle1 (void);
void handle2 (void);
void handle3 (void);
void handle4 (void);
void key_1 (void);
void key_2 (void);
void key_3 (void);
void key_4 (void);
void sensor_1 (void);
void sensor_2 (void);
void sensor_3 (void);
void sensor_4 (void);
void saveEEPROM (void);
void sendPC (void);

void InitScheduler (void)
{
	u8 i;
	// timer prescaler clock/1024
	TCCR0 |= (1<<CS02)|(1<<CS00);
	// clear pending interrupts
	TIFR = 1<<TOV0;
	// enable timer0 overflow interrupt
	TIMSK |= 1<<TOIE0;
	// load timer0
	TCNT0 = StartFrom;

	// clear task array
	for (i=0; i<MAXnTASKS; i++) DeleteTask(i);
}

void DeleteTask (u8 j)
{
	TaskArray[j].pfunc = 0x0000;
	TaskArray[j].delay = 0;
	TaskArray[j].period = 0;
	TaskArray[j].run = 0;
}

void AddTask (void (*taskfunc)(void), u16 taskdelay, u16 taskperiod)
{
	u8 n=0;

	// find next available position
	while ((TaskArray[n].pfunc != 0) && (n < MAXnTASKS)) n++;

	// place task
	if (n < MAXnTASKS)
	{
		TaskArray[n].pfunc = taskfunc;
		TaskArray[n].delay = taskdelay;
		TaskArray[n].period = taskperiod;
		TaskArray[n].run = 0;
	}
}


/// Main //////////////
int main(void)
{
	initVar();
	USART_Init ();
	initKeypad();
	initSensor();
	InitScheduler();
	lcd_init();
```

```
	// populate task array
	AddTask (update_lcd, 0, 0);
	AddTask (sensor_1, 0, 0);
	AddTask (sensor_2, 0, 0);
	AddTask (sensor_3, 0, 0);
	AddTask (sensor_4, 0, 0);
	AddTask (saveEEPROM, 0, 5999);
	AddTask (key_1, 0, 9);
	AddTask (key_2, 0, 9);
	AddTask (key_3, 0, 9);
	AddTask (key_4, 0, 9);

	AddTask (lcd_display, 0 , 4);
	AddTask (sendPC, 0 , 99);
	// enable interrupts
	sei();
	while (1)
	{
		DispatchTask();
	}
}


SIGNAL(SIG_OVERFLOW0)
{
	u8 m;

	// load timer
	TCNT0 = StartFrom;

	for (m=0; m<MAXnTASKS; m++)
	{
		if (TaskArray[m].pfunc)
		{
			if (TaskArray[m].delay == 0)
			{
				TaskArray[m].run = 1;
				TaskArray[m].delay = TaskArray[m].period;
			}
			else TaskArray[m].delay--;
		}
	}
}


void DispatchTask (void)
{
	u8 k;

	for (k=0; k<MAXnTASKS; k++)
	{
		if (TaskArray[k].run == 1)
		{
			// run task
			(*TaskArray[k].pfunc)();
			// clear run flag
			TaskArray[k].run = 0;
		}
	}
}



/// Defines //////////

#define TxbufLength 128
```

```
#define TxbufMask TxbufLength - 1
#define RxbufLength 128
#define RxbufMask RxbufLength - 1


/// Globals ///////////
static u8 TxBuf[TxbufLength];
volatile static u8 TxWR, TxRD, numTx;
static u8 RxBuf[RxbufLength];
volatile static u8 RxWR, RxRD, numRx;

/// Prototypes ////////
void USART_Init(void);
void TransmitByte(u8 data);
void TransmitString(char *);

void USART_Init(void)
{
        /* Set baud rate */
        UBRRH = (unsigned char)(MYUBRR>>8);
        UBRRL = (unsigned char)MYUBRR;

        // enable Rx, Tx interrupts, 0 parity, 1 stop bit
        UCSRB = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
        UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);

        // initialize Tx, Rx buffer pointers
        TxWR = 0; TxRD = 0; numTx = 0;

}

SIGNAL(SIG_UART_RECV)
{
  // advance RxWR
  RxWR++;
  // mask it
  RxWR &= RxbufMask;
  // store received byte in buffer
  RxBuf[RxWR] = UDR;
  // data added
  numRx++;
}

u8 ReceiveByte(void)
{
  // advance RxRD
  RxRD++;
  // mask it
  RxRD &= RxbufMask;
  // data consumed
  numRx--;
  // return data
  return RxBuf[RxRD];
}

SIGNAL(SIG_UART_DATA)
{
  // check if more bytes are available
  if (numTx>0)
  {
    // advance TxRD
    TxRD++;
    // mask it
    TxRD &= TxbufMask;
    // start transmission by loading UDR
    UDR = TxBuf[TxRD];
    // data sent
```

```
    numTx--;
  }
  else
  // Disable UDRE interrupt (no more bytes)
  UCSRB &= ~(1<<UDRIE);
}

void TransmitByte(u8 data)
{
  // advance TxWR
  TxWR++;
  // mask it
  TxWR &= TxbufMask;
  // store data in buffer
  TxBuf[TxWR] = data;
  // data added
  numTx++;
  // enable UDRE interrupt
  UCSRB |= 1<<UDRIE;
}

void TransmitString(char *data)
{
        while(*data != '\0') {
                TransmitByte(*data);
                data++;
        }
}

void initKeypad(void)
{
   DDRA = 0x0F;
   PORTA = 0xFE;
}

void initSensor(void)
{
   DDRD &= 0x0F;
   PORTD |= 0xF0;
}

void initVar (void)
{
        count_time = 0;
        state = 0;
        menuindex = 0;
        savetime = 0;
        rundisplay = 0;
        k1_old = 0;
        k2_old = 0;
        k3_old = 0;
        k4_old = 0;
        s1_old = 0;
        s2_old = 0;
        s3_old = 0;
        s4_old = 0;
        count1 = 0;
        count2 = 0;
        count3 = 0;
        count4 = 0;
        save_count = 0;
        send_count = 0;
}


void update_lcd(void)
{
```

```
		if(lcd_state < 16) {
			lcd_data(lcd_line1[lcd_state]);
		}
		else if(lcd_state == 16) {
			lcd_command(0xC0);
		}
		else if(lcd_state < 33) {
			lcd_data(lcd_line2[lcd_state - 17]);
		}
		else if(lcd_state == 33) {
			lcd_command(0x80);
		}
lcd_state++;
		if(lcd_state == 34) lcd_state = 0;
}


void key_1(void) // Up or A
{
		k1 = bit_is_clear(PINA, key1);
		if((k1 == 1) && (k1_old == 0)) {
			handle1();
		}
		k1_old = k1;
}

void key_2(void) // Down or B
{
		k2 = bit_is_clear(PINA, key2);
		if((k2 == 1) && (k2_old == 0)) {
			handle2();
		}
		k2_old = k2;
}


void key_3(void) // OK or C
{
		k3 = bit_is_clear(PINA, key3);
		if((k3 == 1) && (k3_old == 0)) {
			handle3();
		}
		k3_old = k3;
}

void key_4(void) // menu or D
{
		k4 = bit_is_clear(PINA, key4);
		if((k4 == 1) && (k4_old == 0)) {
			handle4();
		}
		k4_old = k4;
}


void handle1(void)
{
		if(state == MENU) {
			if(menuindex == 1) {
				if(savetime < MAXTIME) savetime++;
			}
		}
		else if(state == RUN) {
			rundisplay = 0;
		}
}
```

```
void handle2(void)
{
	if(state == MENU) {
		if(menuindex == 1) {
			if(savetime > 0) savetime--;
		}
	}
	else if(state == RUN) {
		rundisplay = 1;
	}
}

void handle3(void)
{
	if(state == MENU) {
		if(menuindex == 0) {
			state = SEND_PC;
		}
		else if(menuindex == 2) {
			state = RUN;
		}
	}
	else if(state == RUN) {
		rundisplay = 2;
	}
}

void handle4(void)
{
	if(state == MENU) {
		menuindex++;
		if(menuindex > 2) menuindex = 0;
	}
	else if(state == RUN) {
		rundisplay = 3;
	}
}


void sensor_1(void)
{
	if(state == RUN) {
		s1 = bit_is_clear(PIND, sensor1);
		if((s1 == 1) && (s1_old == 0)) {
			count1++;
		}
		s1_old = s1;
	}
}

void sensor_2(void)
{
	if(state == RUN) {
		s2 = bit_is_clear(PIND, sensor2);
		if((s2 == 1) && (s2_old == 0)) {
			count2++;
		}
		s2_old = s2;
	}
}

void sensor_3(void)
{
	if(state == RUN) {
		s3 = bit_is_clear(PIND, sensor3);
		if((s3 == 1) && (s3_old == 0)) {
			count3++;
```

```
                }
                s3_old = s3;
        }
}

void sensor_4(void)
{
        if(state == RUN) {
                s4 = bit_is_clear(PIND, sensor4);
                if((s4 == 1) && (s4_old == 0)) {
                        count4++;
                }
                s4_old = s4;
        }
}

void eeprom_write_16bit(uint16_t address, uint16_t value)
{
        eeprom_write_byte ((uint8_t *) address, value >> 8);
        eeprom_write_byte ((uint8_t *) (address + 1), (uint8_t) value);
}


uint16_t eeprom_read_16bit(uint16_t address)
{
        uint8_t high, low;
        uint16_t result;
        high = eeprom_read_byte((uint8_t *) address);
        low = eeprom_read_byte((uint8_t *) (address+1));
        result = (uint16_t)(high) * 256 + (uint16_t) (low);
        return result;
}


void saveEEPROM(void)
{
        if(state == RUN)
        {

                if(count1_old == count1) {
                        address = (uint16_t) save_count * 2 + 1;
                        eeprom_write_16bit(address, count1);}

                if(count2_old == count2) {
                        address = (uint16_t) save_count * 2 + 101;
                        eeprom_write_16bit(address, count2);}

                if(count3_old == count3) {
                        address = (uint16_t) save_count * 2 + 201;
                        eeprom_write_16bit(address, count3);}

                if(count4_old == count4) {
                        address = (uint16_t) save_count * 2 + 301;
                        eeprom_write_16bit(address, count4);}

                count_time++;
                if(count_time == time[savetime]) {
                        count_time = 0;

                        address = (uint16_t) save_count * 2 + 1;
                        eeprom_write_16bit(address, count1);

                        address = (uint16_t) save_count * 2 + 101;
                        eeprom_write_16bit(address, count2);

                        address = (uint16_t) save_count * 2 + 201;
                        eeprom_write_16bit(address, count3);
```

```
                        address = (uint16_t) save_count * 2 + 301;
                        eeprom_write_16bit(address, count4);

                        save_count++;
                        eeprom_write_byte((uint8_t *) 512, save_count);
                        if(save_count == 50) {
                                initVar();
                                state = MENU;
                        }
                }
        }
}

void sendPC (void)
{
        if(state == SEND_PC) {
                num_data = eeprom_read_byte((uint8_t *) 512);
                if(num_data < 51) {
                        address = (uint16_t) send_count * 2 + 1;
                        data1 = eeprom_read_16bit(address);

                        address = (uint16_t) send_count * 2 + 101;
                        data2 = eeprom_read_16bit(address);

                        address = (uint16_t) send_count * 2 + 201;
                        data3 = eeprom_read_16bit(address);

                        address = (uint16_t) send_count * 2 + 301;
                        data4 = eeprom_read_16bit(address);


                        sprintf(buff, "%i cm\t\t\t%i cm\t\t\t\t%i cm\t\t\t\t%i cm", data1, data2, data3, data4);
                        TransmitString(buff);


                        send_count++;
                        if(send_count == num_data) {
                                initVar();
                                state = MENU;
                        }
                }
                else {
                        TransmitString("No Data\r\n");
                        state = MENU;
                }
        }
}

void lcd_display(void)
{
        if(state == MENU) {
                switch(menuindex) {
                        case 0:
                                sprintf(lcd_line1, "Send Data to PC  ");
                                sprintf(lcd_line2, "   OK to start       ");
                                break;
                        case 1:
                                sprintf(lcd_line1, "Save Time Period ");
                                sprintf(lcd_line2, "%i minutes          ",time[savetime]);
                                break;
                        case 2:
                                sprintf(lcd_line1, "Measure Length   ");
                                sprintf(lcd_line2, "   OK to start       ");
                                break;
                }
        }
```

```
        else if(state == SEND_PC) {
                sprintf(lcd_line1, "Sending Data     ");
                sprintf(lcd_line2, "...              ");
        }
        else if(state == RUN) {
                switch(rundisplay) {
                        case 0:
                                sprintf(lcd_line1, "Machine A       ");
                                sprintf(lcd_line2, "%i cm          ", count1);
                                break;
                        case 1:
                                sprintf(lcd_line1, "Machine B       ");
                                sprintf(lcd_line2, "%i cm          ", count2);
                                break;
                        case 2:
                                sprintf(lcd_line1, "Machine C       ");
                                sprintf(lcd_line2, "%i cm          ", count3);
                                break;
                        case 3:
                                sprintf(lcd_line1, "Machine D       ");
                                sprintf(lcd_line2, "%i cm          ", count4);
                                break;
                }
        }
}
```

SUBPROGRAM LCD

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/wdt.h>

#include "lcd.h"

** constants/macros

#define DDR(x) (*(&x - 1))  /* address of data direction register of port x */
#define PIN(x) (*(&x - 2))  /* address of input register of port x          */

#define lcd_e_delay()   __asm__ __volatile__( "rjmp 1f\n 1:" );
#define lcd_e_high()   LCD_E_PORT  |=  _BV(LCD_E_PIN);
#define lcd_e_low()    LCD_E_PORT  &= ~_BV(LCD_E_PIN);
#define lcd_rw_high()   LCD_RW_PORT |=  _BV(LCD_RW_PIN)
#define lcd_rw_low()    LCD_RW_PORT &= ~_BV(LCD_RW_PIN)
#define lcd_rs_high()   LCD_RS_PORT |=  _BV(LCD_RS_PIN)
#define lcd_rs_low()    LCD_RS_PORT &= ~_BV(LCD_RS_PIN)

#define LCD_FUNCTION_DEFAULT    LCD_FUNCTION_4BIT_2LINES


/*************************************************************************
 delay loop for small accurate delays: 16-bit counter, 4 cycles/loop
*************************************************************************/
static inline void _delayFourCycles(unsigned int __count)
{
   if ( __count == 0 )
      __asm__ __volatile__( "rjmp 1f\n 1:" );    // 2 cycles
   else
      __asm__ __volatile__ (
               "1: sbiw %0,1" "\n\t"
               "brne 1b"                          // 4 cycles/loop
               : "=w" (__count)
               : "0" (__count)
              );
}


/*************************************************************************
delay for a minimum of <us> microseconds
the number of loops is calculated at compile-time from MCU clock frequency
*************************************************************************/
#define delay(us)  _delayFourCycles( ( ( 1*(XTAL/4000) )*us)/1000 )


/* toggle Enable Pin to initiate write */
static void lcd_e_toggle(void)
{
   lcd_e_high();

   lcd_e_delay();

   lcd_e_low();
}


/*************************************************************************
Low-level function to write byte to LCD controller
Input:    data   byte to write to LCD
          rs     1: write data
                 0: write instruction
Returns:  none
*************************************************************************/
```

```
static void lcd_write(uint8_t data, uint8_t rs)
{
   unsigned char dataBits ;


   if (rs) lcd_rs_high();   /* write data        (RS=1, RW=0) */
   else    lcd_rs_low();    /* write instruction (RS=0, RW=0) */
   lcd_rw_low();

   /* configure data pins as output */
   DDR(LCD_DATA_PORT) |= 0xF0;

   /* output high nibble first */
   dataBits = LCD_DATA_PORT & 0x0F;
   LCD_DATA_PORT = dataBits |(data&0xF0);
   lcd_e_toggle();

   /* output low nibble */
   LCD_DATA_PORT = dataBits | (data<<4);
   lcd_e_toggle();

   /* all data pins high (inactive) */
   LCD_DATA_PORT = dataBits | 0xF0;
}

/*************************************************************************
Low-level function to read byte from LCD controller
Input:    rs     1: read data
               0: read busy flag / address counter
Returns:  byte read from LCD controller
*************************************************************************/
static uint8_t lcd_read(uint8_t rs)
{
   uint8_t data;

   if (rs) lcd_rs_high();
   else    lcd_rs_low();
   lcd_rw_high();

   DDR(LCD_DATA_PORT) &= 0x0F;
   LCD_DATA_PORT |= 0xF0;              /* enable pullups to get a busy */

   lcd_e_high();
   lcd_e_delay();
   data = PIN(LCD_DATA_PORT) & 0xF0;
   lcd_e_low();

   lcd_e_delay();

   lcd_e_high();
   lcd_e_delay();
   data |= PIN(LCD_DATA_PORT) >> 4;
   lcd_e_low();

   return data;
}


/*************************************************************************
loops while lcd is busy, returns address counter
*************************************************************************/
static void lcd_waitbusy(void)
{
  while ( (lcd_read(0)) & (1<<LCD_BUSY));
}

** PUBLIC FUNCTIONS
```

```
/*************************************************************************
Send LCD controller instruction command
Input:   instruction to send to LCD controller, see HD44780 data sheet
Returns: none
*************************************************************************/
void lcd_command(uint8_t cmd)
{
   lcd_write(cmd, 0);
}

void lcd_command_b(uint8_t cmd)
{
   lcd_waitbusy();
   lcd_write(cmd, 0);
}

/*************************************************************************
Send data byte to LCD controller
Input:   data to send to LCD controller, see HD44780 data sheet
Returns: none
*************************************************************************/
void lcd_data(uint8_t data)
{
   lcd_write(data, 1);
}

void lcd_data_d(uint8_t data)
{
   lcd_waitbusy();
   lcd_write(data, 1);
}

/*************************************************************************
Clear display and set cursor to home position
*************************************************************************/
void lcd_clrscr()
{
   lcd_command_b(1<<LCD_CLR);
}


/*************************************************************************
Initialize display
Returns:  0 on failure, 1 else
*************************************************************************/
uint8_t lcd_init()
{
    *  Initialize LCD to 4 bit I/O mode


   /* configure all port bits as output (all LCD lines on same port) */
   DDR(LCD_DATA_PORT) |= 0xF0;

   /* configure all port bits as output (all LCD data lines on same */
   /* port, but control lines on different ports) */
   DDR(LCD_RS_PORT)    |= _BV(LCD_RS_PIN);
   DDR(LCD_RW_PORT)    |= _BV(LCD_RW_PIN);
   DDR(LCD_E_PORT)     |= _BV(LCD_E_PIN);  /* first controller   */

   delay(16000);        /* wait 16ms or more after power-on       */

   /* initial write to lcd is 8bit */
   LCD_DATA_PORT = (LCD_DATA_PORT & 0x0F) | _BV(LCD_FUNCTION) |
     _BV(LCD_FUNCTION_8BIT);
   lcd_e_toggle();
   delay(4992);         /* delay, busy flag can't be checked here */
```

```
    /* repeat last command */
    lcd_e_toggle();
    delay(64);

    /* repeat last command a third time */
    lcd_e_toggle();
    delay(64);

    /* now configure for 4bit mode */
    LCD_DATA_PORT &= ~(_BV(LCD_FUNCTION) >> 4);
    lcd_e_toggle();
    delay(64);

    /* from now the LCD only accepts 4 bit I/O, we can use lcd_command() */

    /* try to find out of there's a controller and return 0 if not */

    /* display must not be busy anymore */
    if((lcd_read(0)) & (1<<LCD_BUSY))
      return 0;

    /* function set: display lines  */
    lcd_command_b(LCD_FUNCTION_DEFAULT);

    /* wait some time */
    delay(64);

    /* display must not be busy anymore */
    if((lcd_read(0)) & (1<<LCD_BUSY))
      return 0;

    lcd_command_b(LCD_DISP_OFF);
    lcd_clrscr();
    lcd_command_b(LCD_MODE_DEFAULT);
    lcd_command_b(LCD_DISP_ON);
    return 1;
}/* lcd_init */

#if (__GNUC__ * 100 + __GNUC_MINOR__) < 303
#error "This library requires AVR-GCC 3.3 or later, update to newer AVR-GCC compiler !"
#endif

#include <inttypes.h>
#include <avr/pgmspace.h>

/**
 *  @name  Definitions for MCU Clock Frequency
 *  Adapt the MCU clock frequency in Hz to your target.
 */
#define XTAL 12000000
/**
 *  @name Definitions for 4-bit IO mode
 *  Change LCD_PORT if you want to use a different port for the LCD pins.
 *
 *  The four LCD data lines and the three control lines RS, RW, E can be on the
 *  same port or on different ports.
 *  Change LCD_RS_PORT, LCD_RW_PORT, LCD_E_PORT if you want the control lines on
 *  different ports.
 *
 *  Normally the four data lines should be mapped to bit 0..3 on one port, but it
 *  is possible to connect these data lines in different order or even on different
 *  ports by adapting the LCD_DATAx_PORT and LCD_DATAx_PIN definitions.
 *
 */
#define LCD_DATA_PORT    PORTC
#define LCD_RS_PORT      PORTC
#define LCD_RS_PIN       0
```

```
#define LCD_RW_PORT      PORTC
#define LCD_RW_PIN       1
#define LCD_E_PORT       PORTC
#define LCD_E_PIN        2

/**
 *  @name Definitions for LCD command instructions
 *  The constants define the various LCD controller instructions which can be passed to the
 *  function lcd_command(), see HD44780 data sheet for a complete description.
 */

/* instruction register bit positions, see HD44780U data sheet */
#define LCD_CLR              0
#define LCD_HOME             1
#define LCD_ENTRY_MODE       2
#define LCD_ENTRY_INC        1
#define LCD_ENTRY_SHIFT      0
#define LCD_ON               3
#define LCD_ON_DISPLAY       2
#define LCD_ON_CURSOR        1
#define LCD_ON_BLINK         0
#define LCD_MOVE             4
#define LCD_MOVE_DISP        3
#define LCD_MOVE_RIGHT       2
#define LCD_FUNCTION         5
#define LCD_FUNCTION_8BIT    4
#define LCD_FUNCTION_2LINES  3
#define LCD_FUNCTION_10DOTS  2
#define LCD_CGRAM            6
#define LCD_DDRAM            7
#define LCD_BUSY             7

/* set entry mode: display shift on/off, dec/inc cursor move direction */
#define LCD_ENTRY_DEC           0x04
#define LCD_ENTRY_DEC_SHIFT     0x05
#define LCD_ENTRY_INC_          0x06
#define LCD_ENTRY_INC_SHIFT     0x07

/* display on/off, cursor on/off, blinking char at cursor position */
#define LCD_DISP_OFF            0x08
#define LCD_DISP_ON             0x0C
#define LCD_DISP_ON_BLINK       0x0D
#define LCD_DISP_ON_CURSOR      0x0E
#define LCD_DISP_ON_CURSOR_BLINK 0x0F

/* move cursor/shift display */
#define LCD_MOVE_CURSOR_LEFT    0x10
#define LCD_MOVE_CURSOR_RIGHT   0x14
#define LCD_MOVE_DISP_LEFT      0x18
#define LCD_MOVE_DISP_RIGHT     0x1C
/* function set: set interface data length and number of display lines */
#define LCD_FUNCTION_4BIT_1LINE  0x20
#define LCD_FUNCTION_4BIT_2LINES 0x28
#define LCD_FUNCTION_8BIT_1LINE  0x30
#define LCD_FUNCTION_8BIT_2LINES 0x38


#define LCD_MODE_DEFAULT     ((1<<LCD_ENTRY_MODE) | (1<<LCD_ENTRY_INC) )



/**
 *  @name Functions
 */


/**
```

```
 @brief    Initialize display and select type of cursor
 @param    dispAttr \b LCD_DISP_OFF display off\n
             \b LCD_DISP_ON display on, cursor off\n
             \b LCD_DISP_ON_CURSOR display on, cursor on\n
             \b LCD_DISP_ON_CURSOR_BLINK display on, cursor on flashing
 @return  0 on failure, 1 on success
*/
extern uint8_t lcd_init(void);


/**
 @brief    Clear display and set cursor to home position
 @param    void
 @return   none
*/
extern void lcd_clrscr(void);

/**
 @brief    Display string without auto linefeed
 @param    s string to be displayed
 @return   none
*/
extern void lcd_puts(const char *s);

/**
 @brief    Send LCD controller instruction command
 @param    cmd instruction to send to LCD controller, see HD44780 data sheet
 @return   none
*/
extern void lcd_command(uint8_t cmd);


/**
 @brief    Send data byte to LCD controller

 Similar to lcd_putc(), but without interpreting LF
 @param    data byte to send to LCD controller, see HD44780 data sheet
 @return   none
*/
extern void lcd_data(uint8_t data);

/*@}*/
#endif //LCD_H
```

LAMPIRAN C
DATASHEET

DATA SHEET

2N2222; 2N2222A
NPN switching transistors

Product specification
Supersedes data of September 1994
File under Discrete Semiconductors, SC04

1997 May 29

Philips Semiconductors

PHILIPS

NPN switching transistors 2N2222; 2N2222A

FEATURES

- High current (max. 800 mA)
- Low voltage (max. 40 V).

APPLICATIONS

- Linear amplification and switching.

DESCRIPTION

NPN switching transistor in a TO-18 metal package.
PNP complement: 2N2907A.

PINNING

PIN	DESCRIPTION
1	emitter
2	base
3	collector, connected to case

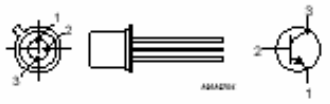


Fig.1 Simplified outline (TO-18) and symbol.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	2N2222		–	60	V
	2N2222A		–	75	V
V_{CEO}	collector-emitter voltage	open base			
	2N2222		–	30	V
	2N2222A		–	40	V
I_C	collector current (DC)		–	800	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
h_{FE}	DC current gain	I_C = 10 mA; V_{CE} = 10 V	75	–	
f_T	transition frequency	I_C = 20 mA; V_{CE} = 20 V; f = 100 MHz			
	2N2222		250	–	MHz
	2N2222A		300	–	MHz
t_{off}	turn-off time	I_{Con} = 150 mA; I_{Bon} = 15 mA; I_{Boff} = –15 mA	–	250	ns

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	2N2222		–	60	V
	2N2222A		–	75	V
V_{CEO}	collector-emitter voltage	open base			
	2N2222		–	30	V
	2N2222A		–	40	V
V_{EBO}	emitter-base voltage	open collector			
	2N2222		–	5	V
	2N2222A		–	6	V
I_C	collector current (DC)		–	800	mA
I_{CM}	peak collector current		–	800	mA
I_{BM}	peak base current		–	200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	500	mW
		$T_{case} \leq 25$ °C	–	1.2	W
T_{stg}	storage temperature		–65	+150	°C
T_j	junction temperature		–	200	°C
T_{amb}	operating ambient temperature		–65	+150	°C

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j-a}$	thermal resistance from junction to ambient	in free air	350	K/W
$R_{th\ j-c}$	thermal resistance from junction to case		146	K/W

CHARACTERISTICS

T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
I_{CBO}	collector cut-off current				
	2N2222	I_E = 0; V_{CB} = 50 V	–	10	nA
		I_E = 0; V_{CB} = 50 V; T_{amb} = 150 °C	–	10	µA
I_{CBO}	collector cut-off current				
	2N2222A	I_E = 0; V_{CB} = 60 V	–	10	nA
		I_E = 0; V_{CB} = 60 V; T_{amb} = 150 °C	–	10	µA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = 3 V	–	10	nA
h_{FE}	DC current gain	I_C = 0.1 mA; V_{CE} = 10 V	35	–	
		I_C = 1 mA; V_{CE} = 10 V	50	–	
		I_C = 10 mA; V_{CE} = 10 V	75	–	
		I_C = 150 mA; V_{CE} = 1 V; note 1	50	–	
		I_C = 150 mA; V_{CE} = 10 V; note 1	100	300	
h_{FE}	DC current gain	I_C = 10 mA; V_{CE} = 10 V; T_{amb} = –55 °C			
	2N2222A		35	–	
h_{FE}	DC current gain	I_C = 500 mA; V_{CE} = 10 V; note 1			
	2N2222		30	–	
	2N2222A		40	–	
V_{CEsat}	collector-emitter saturation voltage				
	2N2222	I_C = 150 mA; I_B = 15 mA; note 1	–	400	mV
		I_C = 500 mA; I_B = 50 mA; note 1	–	1.6	V
V_{CEsat}	collector-emitter saturation voltage				
	2N2222A	I_C = 150 mA; I_B = 15 mA; note 1	–	300	mV
		I_C = 500 mA; I_B = 50 mA; note 1	–	1	V
V_{BEsat}	base-emitter saturation voltage				
	2N2222	I_C = 150 mA; I_B = 15 mA; note 1	–	1.3	V
		I_C = 500 mA; I_B = 50 mA; note 1	–	2.6	V
V_{BEsat}	base-emitter saturation voltage				
	2N2222A	I_C = 150 mA; I_B = 15 mA; note 1	0.6	1.2	V
		I_C = 500 mA; I_B = 50 mA; note 1	–	2	V
C_c	collector capacitance	I_E = I_e = 0; V_{CB} = 10 V; f = 1 MHz	–	8	pF
C_e	emitter capacitance	I_C = I_c = 0; V_{EB} = 500 mV; f = 1 MHz			
	2N2222A		–	25	pF
f_T	transition frequency	I_C = 20 mA; V_{CE} = 20 V; f = 100 MHz			
	2N2222		250	–	MHz
	2N2222A		300	–	MHz
F	noise figure	I_C = 200 µA; V_{CE} = 5 V; R_S = 2 kΩ; f = 1 kHz; B = 200 Hz			
	2N2222A		–	4	dB

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
Switching times (between 10% and 90% levels); see Fig.2					
t_{on}	turn-on time	I_{Con} = 150 mA; I_{Bon} = 15 mA; I_{Boff} = −15 mA	–	35	ns
t_d	delay time		–	10	ns
t_r	rise time		–	25	ns
t_{off}	turn-off time		–	250	ns
t_s	storage time		–	200	ns
t_f	fall time		–	60	ns

LAMPIRAN D
DATASHEET SENSOR PROXIMITY

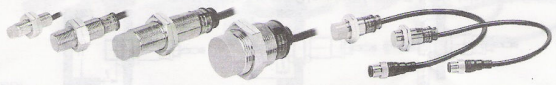
Autonics

INDUCTIVE PROXIMITY SENSOR

CYLINDRICAL TYPE DC 3WIRE

M A N U A L

CE



Thank you very much for selecting Autonics products.
For your safety, please read the following before using.

▣ Caution for your safety

※Please keep these instructions and review them before using this unit.

※Please observe the cautions that follow;

⚠ **Warning** Serious injury may result if instructions are not followed.

⚠ **Caution** Product may be damaged, or injury may result if instructions are not followed.

※The following is an explanation of the symbols used in the operation manual.
⚠caution:Injury or danger may occur under special conditions.

⚠ Warning

1. **In case of using this unit with machineries(Nuclear power control, medical equipment vehicle, train, airplane, combustion apparatus, entertainment or safety device etc), it requires installing fail-safe device, or contact us for information on type required.**
   It may result in serious damage, fire or human injury.

⚠ Caution

1. **Do not use this unit in place where there are flammable, explosive gas, chemical or strong alkalis, acids.**
   It may cause a fire or explosion.
2. **Do not impact on this unit.**
   It may result in malfunction or damage to the product.
3. **Do not apply AC power and observe specification rating.**
   It may result in serious damage to the product.

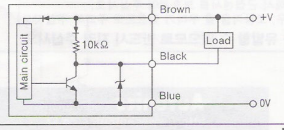
▣ Ordering information

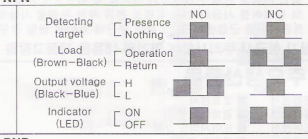
P R L 18 – 5 DN

- DC 3 wire output
  - DN : NPN NO(Normal Open)
  - DN2 : NPN NC(Normal Close)
  - DP : PNP NO(Normal Open)
  - DP2 : PNP NC(Normal Close)
- Standard detecting distance(mm)
- Diameter of head(mm)
- Type
  - No indication: Standard
  - S : Short pipe type
  - L : Long pipe type
  - W : Outgoing connector cable type
  - WL: Outgoing connector cable type with long pipe
- Appearance
  - Cylindrical type
  - Inductive proximity sensor

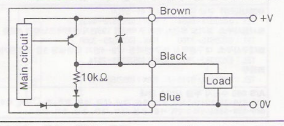
▣ Control output diagram & Load operating

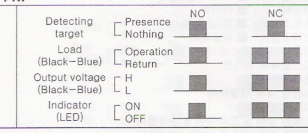
NPN





PNP



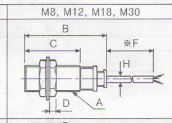
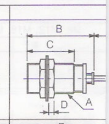
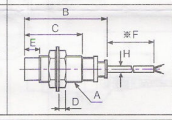
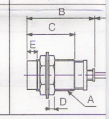


※**The above specifications are changeable at anytime without notice.**

▣ Specifications

Model	PR08-1.5DN PR08-1.5DP PR08-1.5DN2 PR08-1.5DP2 PRL08-1.5DN PRL08-1.5DP PRL08-1.5DN2 PRL08-1.5DP2 PRW08-1.5DN PRW08-1.5DP PRW08-1.5DN2 PRW08-1.5DP2 PRWL08-1.5DN PRWL08-1.5DP PRWL08-1.5DN2 PRWL08-1.5DP2	PR08-2DN PR08-2DP PR08-2DN2 PR08-2DP2 PRL08-2DN PRL08-2DP PRL08-2DN2 PRL08-2DP2 PRW08-2DN PRW08-2DP PRW08-2DN2 PRW08-2DP2 PRWL08-2DN PRWL08-2DP PRWL08-2DN2 PRWL08-2DP2	PR12-2DN PR12-2DP PR12-2DN2 PR12-2DP2 PRS12-2DN PRS12-2DP PRS12-2DN2 PRW12-2DN PRW12-2DP PRW12-2DN2 PRW12-2DP2	PR12-4 PR12-4 PR12-4 PR12-4 PRS12- PRS12- PRS12- PRW12- PRW12- PRW12- PRW12-
Detecting distance	1.5mm ±10%	2mm ±10%	2mm ±10%	4mm
Hysteresis	Max.			
Standard detecting target	8×8×1mm (Iron)		12×12×1mm (Iron)	
Setting distance	0 to 1.05	0 to 1.4		0 to
Power supply (Operating voltage)				
Current consumption				
Response frequency	800Hz			400
Residual voltage	Max. 2.0V			
Affection by Temp.	±10% max. of detecting distance at +20℃ withi			
Control output				
Insulation resistance				
Dielectric strength				150
Vibration	1mm amplitude at frequency of			
Shock	500m/s² (5			
Indicator	Ope			
Ambient temperature	−25 to +			
Storage temperature	−30 to +			
Ambient humidity				
Protection circuit	Reverse polarity protection, su			
Protection	I			
Weight	PR:Approx. 68g PRL:Approx. 70g PRW:Approx. 30g PRWL:Approx. 32g		PR:Approx. 70g PRW:Approx. 40g PRS:Approx. 68g	

▣ Dimensions

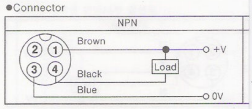
Type	M8, M12, M18, M30	
Flush		
Non-Flush		

Type			A	B	C	D
Flush	M8	PR	M8×1	30	30	4
		PRL	M8×1	40	40	4
		PRW	M8×1	30	30	4
		PRWL	M8×1	40	40	4
	M12	PR	M12×1	42.5	31.5	4
		PRS	M12×1	35.5	24.5	4
		PRW	M12×1	42.5	31.5	4
	M18	PR	M18×1	47	29	4
		PRL	M18×1	80	62	4
		PRW	M18×1	47	29	4
		PRWL	M18×1	80	62	4
	M30	PR	M30×1.5	58	38	5
		PRL	M30×1.5	80	60	5
		PRW	M30×1.5	58	38	5
		PRWL	M30×1.5	80	60	5
Non-Flush	M8	PR	M8×1	30	30	4
		PRL	M8×1	40	40	4
		PRW	M8×1	30	30	4
		PRWL	M8×1	40	40	4
	M12	PR	M12×1	42.5	31.5	4
		PRS	M12×1	35.5	24.5	4
		PRW	M12×1	42.5	31.5	4
	M18	PR	M18×1	47	29	4
		PRL	M18×1	80	62	4
		PRW	M18×1	47	29	4
		PRWL	M18×1	80	62	4
	M30	PR	M30×1.5	58	38	5
		PRL	M30×1.5	80	60	5
		PRW	M30×1.5	58	38	5
		PRWL	M30×1.5	80	60	5

※"F" standard : Cable outgoing connector type 300mm.

▣ Connections

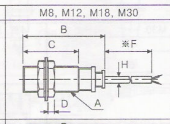
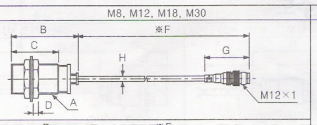
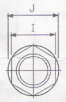
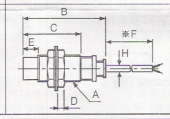
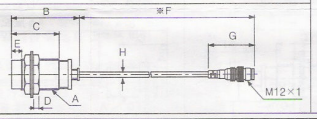
●Connector



▣ Specifications

Model	PR08-1.5DN PR08-1.5DP PR08-1.5DN2 PR08-1.5DP2 PRL08-1.5DN PRL08-1.5DP PRL08-1.5DN2 PRL08-1.5DP2 PRW08-1.5DN PRW08-1.5DP PRW08-1.5DN2 PRW08-1.5DP2 PRWL08-1.5DN PRWL08-1.5DP PRWL08-1.5DN2 PRWL08-1.5DP2	PR08-2DN PR08-2DP PR08-2DN2 PR08-2DP2 PRL08-2DN PRL08-2DP PRL08-2DN2 PRL08-2DP2 PRW08-2DN PRW08-2DP PRW08-2DN2 PRW08-2DP2 PRWL08-2DN PRWL08-2DP PRWL08-2DN2 PRWL08-2DP2	PR12-2DN PR12-2DP PR12-2DN2 PR12-2DP2 PRS12-2DN PRS12-2DP PRS12-2DN2 PRW12-2DN PRW12-2DP PRW12-2DN2 PRW12-2DP2	PR12-4DN PR12-4DP PR12-4DN2 PR12-4DP2 PRS12-4DN PRS12-4DP PRS12-4DN2 PRW12-4DN PRW12-4DP PRW12-4DN2 PRW12-4DP2	PR18-5DN PR18-5DP PR18-5DN2 PR18-5DP2 PRL18-5DN PRL18-5DP PRL18-5DN2 PRL18-5DP2 PRW18-5DN PRW18-5DP PRW18-5DN2 PRW18-5DP2 PRWL18-5DN PRWL18-5DP PRWL18-5DN2 PRWL18-5DP2	PR18-8DN PR18-8DP PR18-8DN2 PR18-8DP2 PRL18-8DN PRL18-8DP PRL18-8DN2 PRL18-8DP2 PRW18-8DN PRW18-8DP PRW18-8DN2 PRW18-8DP2 PRWL18-8DN PRWL18-8DP PRWL18-8DN2 PRWL18-8DP2	PR30-10DN PR30-10DP PR30-10DN2 PR30-10DP2 PRL30-10DN PRL30-10DP PRL30-10DN2 PRL30-10DP2 PRW30-10DN PRW30-10DP PRW30-10DN2 PRW30-10DP2 PRWL30-10DN PRWL30-10DP PRWL30-10DN2 PRWL30-10DP2	PR30-15DN PR30-15DP PR30-15DN2 PR30-15DP2 PRL30-15DN PRL30-15DP PRL30-15DN2 PRL30-15DP2 PRW30-15DN PRW30-15DP PRW30-15DN2 PRW30-15DP2 PRWL30-15DN PRWL30-15DP PRWL30-15DN2 PRWL30-15DP2
Detecting distance	1.5mm ±10%	2mm ±10%	2mm ±10%	4mm ±10%	5mm ±10%	8mm ±10%	10mm ±10%	15mm ±10%
Hysteresis	Max. 10% of detecting distance							
Standard detecting target	8×8×1mm (Iron)		12×12×1mm (Iron)		18×18×1mm (Iron)	25×25×1mm (Iron)	30×30×1mm (Iron)	45×45×1mm (Iron)
Setting distance	0 to 1.05	0 to 1.4		0 to 2.8	0 to 3.5	0 to 5.6	0 to 7	0 to 10.5
Power supply (Operating voltage)	12-24VDC (10-30VDC)							
Current consumption	Max. 10mA							
Response frequency	800Hz			400Hz	350Hz	200Hz	250Hz	100Hz
Residual voltage	Max. 2.0V		Max. 1.5V					
Affection by Temp.	±10% max. of detecting distance at +20℃ within temperature range of −25 to +70℃ (PR□08 series : ±20% max.)							
Control output	Max. 200mA							
Insulation resistance	Min. 50MΩ (500VDC)							
Dielectric strength	1500VAC 50/60Hz for 1minute							
Vibration	1mm amplitude at frequency of 10 to 55Hz in each of X, Y, Z directions for 2 hours							
Shock	500m/s² (50G) X, Y, Z directions for 3 times							
Indicator	Operating indicator : Red LED							
Ambient temperature	−25 to +70℃ (non−freezing condition)							
Storage temperature	−30 to +80℃ (non−freezing condition)							
Ambient humidity	35 to 95%RH							
Protection circuit	Reverse polarity proteciton, surge protection, overload & short circuit protection							
Protection	IP67 (IEC specification)							
Weight	PR:Approx. 68g PRL:Approx. 70g PRW:Approx. 30g PRWL:Approx. 32g		PR:Approx. 70g PRW:Approx. 40g PRS:Approx. 68g		PR:Approx. 119g PRL :Approx. 149g PRW:Approx. 84g PRWL:Approx. 108g	PR:Approx. 118g PRL:Approx. 142g PRW:Approx. 84g PRWL:Approx. 108g	PR:Approx. 184g PRL:Approx. 222g PRW:Approx. 143g PRWL:Approx. 178g	PR:Approx. 181g PRL:Approx. 227g PRW:Approx. 143g PRWL:Approx. 178g

▣ Dimensions

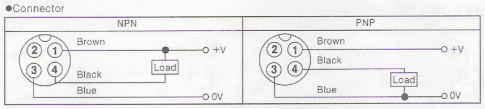
Type	M8, M12, M18, M30	M8, M12, M18, M30	Nut & Washer
Flush			
Non−Flush			

Type			A	B	C	D	E	F	G	H	I	J
Flush	M8	PR	M8×1	30	30	4	——	2000	——	3.5	13	15
		PRL	M8×1	40	40	4	——	2000	——	3.5	13	15
		PRW	M8×1	30	30	4	——	300	44	4	13	15
		PRWL	M8×1	40	40	4	——	300	44	4	13	15
	M12	PR	M12×1	42.5	31.5	4	——	2000	——	4	17	20.5
		PRS	M12×1	35.5	24.5	4	——	2000	——	4	17	20.5
		PRW	M12×1	42.5	31.5	4	——	300	44	4	17	20.5
	M18	PR	M18×1	47	29	4	——	2000	——	5	24	29
		PRL	M18×1	80	62	4	——	2000	——	5	24	29
		PRW	M18×1	47	29	4	——	300	44	5	24	29
		PRWL	M18×1	80	62	4	——	300	44	5	24	29
	M30	PR	M30×1.5	58	38	5	——	2000	——	5	35	42
		PRL	M30×1.5	80	60	5	——	2000	——	5	35	42
		PRW	M30×1.5	58	38	5	——	300	44	5	35	42
		PRWL	M30×1.5	80	60	5	——	300	44	5	35	42
Non−Flush	M8	PR	M8×1	30	30	4	4	2000	——	3.5	13	15
		PRL	M8×1	40	40	4	4	2000	——	3.5	13	15
		PRW	M8×1	30	30	4	4	300	44	4	13	15
		PRWL	M8×1	40	40	4	4	300	44	4	13	15
	M12	PR	M12×1	42.5	31.5	4	7	2000	——	4	17	20.5
		PRS	M12×1	35.5	24.5	4	7	2000	——	4	17	20.5
		PRW	M12×1	42.5	31.5	4	7	300	44	4	17	20.5
	M18	PR	M18×1	47	29	4	10	2000	——	5	24	29
		PRL	M18×1	80	62	4	10	2000	——	5	24	29
		PRW	M18×1	47	29	4	10	300	44	5	24	29
		PRWL	M18×1	80	62	4	10	300	44	5	24	29
	M30	PR	M30×1.5	58	38	5	10	2000	——	5	35	42
		PRL	M30×1.5	80	60	5	10	2000	——	5	35	42
		PRW	M30×1.5	58	38	5	10	300	44	5	35	42
		PRWL	M30×1.5	80	60	5	10	300	44	5	35	42

※"F" standard : Cable outgoing connector type 300mm. (Unit:mm)

▣ Connections

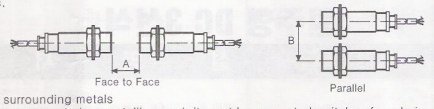
●Connector



▣ Mutual-interference & Influence by surrounding metals

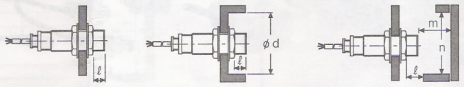
◎Mutual-interference

When plural proximity sensors are mounted in a close row, malfunction of sensor may be caused due to mutual interference. Therefore, be sure to provide a minimum distance between the two sensors, as below charts.



◎Influence by surrounding metals

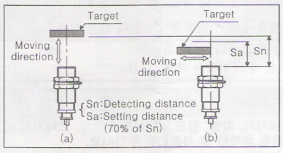
When sensors are mounted on metallic panel, it must be prevented switches from being affected by any metallic object except target. Therefore, be sure to provide a minimum distance as below chart.



(Unit:mm)

Model / Item	PR□08-1.5D□	PR□08-2D□	PR□12-2D□	PR□12-4D□	PR□18-5D□ PRW□18-5D□	PR□18-8D□ PRW□18-8D□	PR□30-10D□ PRW□30-10D□	PR□30-15D□ PRW□30-15D□
A	9	12	12	24	30	48	60	90
B	16	24	24	36	36	54	60	90
ℓ	0	8	0	11	0	14	0	15
ø d	8	12	12	36	18	54	30	90
m	4.5	6	6	12	15	24	30	54
n	12	18	18	36	27	54	45	90

▣ Setting distance



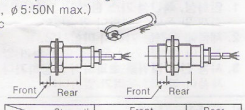
●Detecting distance can be changed by the shape, size or material of the target.
Therefore please check the detecting distance like (a), then pass the target within range of setting distance(Sa).

●Setting distance(Sa)
= Detecting distance(Sn) × 70%
ex)PR30-10DN(See ordering infomation)
Setting distance(Sa) = 10mm × 0.7 = 7mm

▣ Caution for using

1. This equipment shall not be used outdoors or beyond specified temperature range.
2. Do not load over than tensile strength of cord. (ø4:30N max., ø5:50N max.)
3. Do not use the same conduit with cord of this unit and electric power line or power line. Also avoid the same connection.
4. Do not put overload to tighten nut, please use washer for tightening.
   Note1)Allowable strength may be different by the length of head. As see the picture, allowable tightening strength of front part and rear part are in (Chart 1). Rear part includes head nut as like picture.
   Note2)(Chart 1) is for using washer.
5. Please check the voltage changes of power source in order not to excess rating power input.
6. Do not use this unit during transient time(80ms) after apply power.
7. It might result in damage to this product, if use automatic transformer. So please use insulated transformer.
8. Please make wire short as much as possible in order to avoid noise.
9. Be sure to cable as indicated specification on this product. If use wrong cable or bended cable, it shall not maintain the water- proof.
10. It is possible to extend cable with over 0.3mm² and max. 200m.
11. If the target is plated, the operating distance can be changed by the plating material.
12. It may result in malfunction by metal particle on product.
13. If there are machines(Motor, Welding machinery), which occurs big surge around this unit, please install the Varistor or absorber to source of surge, even though there is built-in surge absorber in this unit.
14. If connect the load with big inrush current(DC type bulb) to this unit, the big inrush current will flow due to the initial resistance is low. If the current flows, the resistance of load will be bigger, then it will return to standard current. In this case, proximity sensor might be damaged by inrush current. If you use DC type bulb, please connect extra relay or resistance in order to protect proximity sensor from.
15. If make a transceiver close to proximity sensor or wire connection, it may cause malfunction.

※It may cause malfunction if above instructions are not followed.



Model \ Strength		Size	Front Torque	Rear Torque
PR08 Series	Flush	7mm	90kgf·cm	120kgf·cm
	Non-flush	5mm		
PR12 Series	Flush	13mm	65kgf·cm	120kgf·cm
	Non-flush	7mm		
PR18 Series	Flush	——	150kgf·cm	
	Non-flush			
PR30 Series	Flush	26mm	500kgf·cm	800kgf·cm
	Non-flush	12mm		

(Chart1)

▣ Major products

■ PROXIMITY SENSOR ■ PHOTOELECTRIC SENSOR ■ AREA SENSOR
■ FIBER OPTIC SENSOR ■ DOOR/DOOR SIDE SENSOR ■ PRESSURE SENSOR
■ ROTARY ENCODER ■ SENSOR CONTROLLER
■ SWITCHING POWER SUPPLY
■ TEMPERATURE CONTROLLER
■ TEMPERATURE/HUMIDITY TRANSDUCER
■ POWER CONTROLLER■ RECORDER
■ TACHOMETER/PULSE(RATE) METER
■ PANEL METER ■ INDICATOR
■ SIGNAL CONVERTOR
■ COUNTER ■ TIMER
■ DISPLAY UNIT
■ GRAPHIC PANEL
■ STEPPING MOTOR & DRIVER & MOTION CONTROLLER
■ LASER MARKING SYSTEM(CO2, Nd:YAG)

Autonics Corporation
http://www.autonics.com
Satisfiable Partner For Factory Automation
■HEAD QUARTERS :
41-5, Yongdang-dong, Yangsan-si, Gyeongnam, 626-847, Korea
■OVERSEAS SALES :
Bldg. 402 3rd Fl., Bucheon Techno Park, 193, Yakdae-dong, Wonmi-gu, Bucheon-si, Gyeonggi-do, 420-734, Korea
TEL:82-32-610-2730 / FAX:82-32-329-0728
■E-mail : sales@autonics.com

EP-KE-07-0360D