Appendix A - Programming Information

Introl-CODE functions

As mentioned many times, the C language does not really support many functions intrinsic to the language, but allows for extensions in the C libraries. The following section outlines the most useful functions that are available with the *Introl-CODE* compiler. Some of the functions are ANSI C standard functions, and some are specific to the implementation on the 68HC11. Below is an outline of the structure of this section, along with an example of each section.

Name
: lists the name of the function

Prototype
: shows the prototype of the function which is necessary to know what types are used in the function call and what include file contains the prototype

Description
: describes the use and operation of the function, including options

Portability
: contains the information about the function as applied to other C compilers

Example program
: a short example of how the function is called or used

Related functions
: a list of other functions that are used for the same purpose or that are used with the function being described

abs

Prototype

#include <stdlib.h>
int abs(number);
int number;

Description

abs returns the absolute value of an integer.

Portability

Available on most systems.

Example program

```
/* Absolute value test program
    This program demonstrates the use of the abs function.
*/

/* Include files */
#include <stdlib.h>
main()
{
    int i;

    i = -19;

    printf("\n The absolute value of %d is %d \n\n\n",i,
           abs(i));
}
```

Related functions

fabs

ceil

Prototype

#include <math.h>
double ceil(Number);
double Number;

Description

ceil() is called the ceiling function. It returns the smallest integer greater than Number as a floating point number. For example, ceil(9.3) returns 10.0, and ceil(-6.7) returns -6.0.

Portability

ANSI C compatible

Example program

```
/* Ceiling function test program
    This program demonstrates the use of the ceil function
*/

/* Include files */
#include <math.h>

main()
{
    double i;

    i = -17.569;

    printf("\n The smallest integer greater than %f is \
            %f \n\n",i,ceil(i));
}
```

Related functions

floor

isalpha, isdigit, islower, isspace, isupper

Prototype

int isalpha(Character);
int Character;
...

Description

These functions classify integer values by the ascii characters they represent. Each of these functions returns a non-zero value if the condition is true and a zero value if the condition is false.

isalpha is Character a letter?
isdigit is Character a digit (0 - 9)?
islower is Character a lowercase letter (a - z)?
isupper is Character an uppercase letter (A - Z)?
isspace is Character a space, tab, or carriage return?

Portability

These functions may be defined differently on different systems. Also many similar functions may be available.

Example program

```
/* Character classification test program
    This program demonstrates the use of the character classification functions
*/

main()
{
int i;

i = 113;

if (isalpha(i))
    printf("\n %c is a letter \n",i);
else
    printf(" %c is not a letter \n",i);

if (isdigit(i))
    printf(" %c is a digit \n",i);
else
    printf(" %c is not a digit \n",i);

if (islower(i))
    printf(" %c is a lowercase character \n",i);
else
    printf(" %c is not a lowercase character \n",i);

if (isupper(i))
    printf(" %c is a uppercase character \n",i);
else
    printf(" %c is not a uppercase character \n\n\n",i);
}
```

Related functions

iscntrl, isascii

exp, log, log10

Prototype

#include <math.h>

double exp(x);
double x;

double log(x);
double x;

double log10(x);
double x;

Description

exp returns the exponential of x (e^x)

log returns the natural logarithm of x. x cannot be negative.

log10 returns the common logarithm (base 10) of x. x cannot be negative

Portability

Available on most systems.

Example program

```
/* Logarithm and exponential test program
    This program demonstrates the use of logarithm and exponential functions
*/

/* Include files */
#include <math.h>

main()
{
double x;

x = 0.5153;

printf("\n X is %f \n");
printf(" The natural log of x is %f  \n",log(x));
printf(" The common log of x is %f   \n",log10(x));
printf("  x  \n");
printf(" e  =  %f  \n\n\n",exp(x));
}
```

Related functions

sin, cos, pow, sqrt

pow, sqrt

Prototype

```
#include <math.h>

double pow(x, y);
double x;
int y;

double sqrt(x);
double x;
```

Description

pow returns the number x raised to the power y. x and y cannot both be zero. If x is negative or zero, y must be an integer.

sqrt returns the square root of the number x. x cannot be zero or negative.

Portability

Available on most systems.

Example program

```
/* Power and square root test program
    This program demonstrates the use of pow and sqrt functions
*/

/* Include files */
#include <math.h>

main()
{
double x, y;

x = 2.0;
y = 16.0;

printf("\n X is %f \n",x);
printf(" Y is %f \n\n",y);

printf("  y \n");
printf(" x  = %f\n\n",pow(x,y));

printf(" The square root of x is %f \n\n\n",sqrt(x));
}
```

Related functions

exp, log, log10, sin, cos

fabs

Prototype

#include <math.h>

double fabs(number);
double number;

Description

abs returns the absolute value of a floating point number.

Portability

Available on most systems.

Example program

```
/* Absolute value test program
    This program demonstrates the use of the fabs function.
*/

/* Include files */
#include <math.h>

main()
{
double y;

y = -17.414214;

printf("\n The absolute value of %d is %d \n\n\n",y,fabs(y));
}
```

Related functions

abs

floor

Prototype

```
#include <math.h>

double floor(Number);
double Number;
```

Description

floor returns the greatest integer less than Number as a floating point number. For example floor(9.3) returns 9.0, and floor(-6.7) returns -7.0.

Portability

ANSI C compatible

Example program

```
/* Floor function test program
    This program demonstrates the use of the floor function
*/

/* Include files */
#include <math.h>

main()
{
double i;

i = 14.378;

printf("\n The smallest integer less than %f is %f \n\n",i, floor(i));
}
```

Related functions

ceil

getchar

Prototype

int getchar();
int getc(FILE **stream*);

Description

Getchar reads the next character from the terminal port and returns its value as an integer.

Portability

Getchar is available in ANSI C, but its exact behavior is compiler dependent.

Example program

```
/* getchar test program
    This program demonstrates the use of the getchar function.  The function waits for a character
to be sent from the terminal, then transmits a message with the character back.
*/

main()
{
    int c;

    while (1)
    {
       while(!(c = getchar()));
       printf("The character was %c and it's ASCII code \
               is %d \n",c,c);
    }
}
```

Related functions

gets, putchar

gets

Prototype

#include <string.h>

car *gets(String);
car *String;

Description

Gets will read a line of input from the terminal port and will place it in the string pointed to by String.

Portability

Available everywhere.

Example program

```
/* gets test program
    This program illustrates the use of the gets function.  It will receive a line of input from
the terminal, and return it.  It will also print out the first 40 characters in columns to illustrate
simple string handling functions.  The printf function can only output a string of up to 80 char-
acters, so we check for a string that is too long.  If you need to output longer strings use the
puts function instead.   */

#include <string.h>

main()
{
char String[80];
int i, length;

while (1)
    {
    printf("\nPlease enter a string\n");
    gets(String);

    length = strlen(String);

    if (length <= 60)
        {
        printf("\nThe String is \"%s\" \n",String);

        for (i=0;( (i<length) && (i<40) );i++)
            printf("\t%c",String[i]);

        printf("\n");
        }
    else
        printf("\nThat string is too long, try again.\n");
    }
}
```

Related functions

getchar, puts

putchar, puts

Prototype

void putchar(Character);
int Character;

void puts(String);
char *String;

Description

Putchar sends Character to the terminal port. Character is the ASCII code (integer) for the character to be sent. Puts sends a string to the terminal port.

Portability

ANSI C compatible.

Example program

```
/* putchar & puts test program
    This program illustrates the use of the putchar and puts functions.  The program will use putchar
to output the ASCII characters in the range 33-125 (the printable characters), then print a message
with the puts function.
*/

main()
{
int i;
char String[30];

/* use strcpy() to copy Done message into String */
strcpy(String,"\nDone\n\n\n\n");  puts("\n");

for (i=33;i<=126;i++)
    putchar(i);

puts(String);
}
```

Related functions

gets, getchar

sin, cos, tan, cotan, asin, acos, atan

Prototype

#include <math.h>

double sin(x);
double x;

double cos(x);
double x;

double tan(x);
double x;

double cotan(x);
double x;

double asin(x);
double x;

double acos(x);
double x;

double atan(x);
double x;

Description

These functions perform the appropriate operation on *x* and return the result. *x* is assumed to be in radians.

Portability

Available on most systems.

Example program

```
/* Trigonometric functions test program
    This program demonstrates the use of the trigonometric functions
*/

/* Include files */
#include <math.h>

main()
{
double x;

x = 0.7453;

printf("\n x is %f (Radians)\n");
printf(" The sine of x is %f  \n",sin(x));
printf(" The cosine of x is %f   \n",cos(x));
printf(" The tangent of x is  %f  \n\n\n",tan(x));
}
```

Related functions

exp, log, log10, pow, sqrt

printf

Prototype

```
#include <stdio.h>
int printf(controlString [, arg1] . . . );
char *controlString;
```

Description

The printf function sends formatted output to the serial port of the 68HC11. For more information, refer to a C programing reference book.

The formatted input and output functions are reminiscent of the days of hardcopy terminals, which prompted for input from the user in a specific form, and then output information, one line at a time. This is a very simple method of communication in contrast to today's interfaces that usually have full-screen feedback of keyboard and mouse input. This simple interface can be used with the 68HC11 because the Terminal program emulates a terminal to interface with other computers. Printf is used to send information, such as variables, that must be converted to a character string form before sending each character to the terminal. To accomplish this conversion and positioning on the line, printf uses a control string, which instructs the function what types of input are being used and what format the output should be in. The control string flags are listed here along with several examples of the usage of control strings.

When the printf function searches through the control string argument, it looks for flags in the string that indicate that a command follows. There are two characters which are used: a backslash (\) to indicate a control command follows, and a percent sign (%) to indicate the type of the next variable to send to the terminal.

The control commands follow:

\n	start a new line
\r	send a carriage return without a line feed
\t	send a tab
\	control string continues on the following line

The data types and how to print them are included below:

char	%c
signed char	%d or %c
unsigned char	%d or %c
hexadecimal	%x
short int	%hd
int	%d
long int	%ld
unsigned short int	%hu
unsigned int	%u
unsigned long int	%lu
float, double	%[width].[precision]f

Example - %3.2f prints a floating point or double value with a minimum field width of three digits and two digits after the decimal.

Portability

printf is a standard library function available on all ANSI and original C compilers.

Example program

Related functions

scanf

rand

Prototype

#include <stdlib.h>
int rand(void);

Description

Returns a pseudo-random integer between 0 and 32767. The **srand()** function may be used to seed the pseudo-random number generator before calling **rand()**.

Portability

Available on most systems.

Example program

```
/* rand test program
    This program illustrates the generation of 5 random numbers*/

#include<stdlib.h>

main()
{
int num,i;

for (i=0;i<5;i++)
{
    num=rand();
    printf("the random number=%d\n", num);
}

}
```

Related functions

srand

scanf

Prototype

#include <stdio.h>
int scanf(controlString [, pointer1] . . .);
char *controlString;

Description

Reads formatted data from `stdin` and writes the results to memory at the addresses given by the variable arguments. Each variable argument must be a pointer to a datum of type that corresponds to the format of the data.

Portability

scanf is a standard library function available on all ANSI and original C compilers

Example program

```
#include <stdlib.h>
#include<stdio.h>
void main(void)
{
int a;
printf("Enter an integer:");
scanf("%d", &a);
printf("The value you entered is %d\n", a);
}
```

Related functions

printf

Note: Do not use a "\n" in a scanf statement, as this will cause an error in the compiler. For example:

```
scanf("%d\n",&a);
```

will not work properly.

srand

Prototype

#include <stdlib.h>
int srand(int val);

Description

Seed the **rand()** function with **val**.

Portability

Available on most systems.

Example program

```
/* srand test program
    This program demonstrates how to seed the random number generator*/

#include<stdlib.h>

main()
{
int num;

srand(50);
num=rand();


}
```

Related functions

rand

_VECTOR

Prototype

```
#include <EVB.h>
_VECTOR(FunctionName, VectorNumber);
__mod2__ void FunctionName();
int VectorNumber;
```

Description

VECTOR allows the user to assign a function to a particular interrupt. When the interrupt indicated by VectorNumber occurs, the function FunctionName will be called. Below is a list of 68HC11 interrupts and their corresponding vector numbers.

Vector Description

0	SCI serial system
1	SPI serial transfer complete
2	Pulse accumulator input edge*
3	Pulse accumulator overflow*
4	Timer overflow*
5	Timer output compare 5
6	Timer output compare 4
7	Timer output compare 3
8	Timer output compare 2
9	Timer output compare 1
10	Timer input capture 3
11	Timer input capture 2
12	Timer input capture 1
13	Real time interrupt (used for general timer functions)*
14	IRQ (external pin or parallel I/O)
15	XIRQ pin*+
16	SWI
17	Illegal opcode trap
18	COP failure (reset)
19	COP clock monitor fail (reset)

*These are the most common interrupts for embedded control

+Must be used with a pull-up resistor of approximately 1 kΩ to 5V

Portability

A function similar to this should be available on any embedded control programming system that supports interrupts. You are not likely to find this function in a higher level compiler. It is not defined in ANSI C.

Example program

```
/* Interrupt test program
    This program waits for an IRQ interrupt (14) to occur.  This interrupt can be caused by putting
a low signal on EVB pin #19.  Each time this happens, the function InterruptHandler is called and
the message is printed.  The interrupt will be sent repeatedly as long as there is a low signal on
pin #19
*/
#include <EVB.h>

/* function prototypes */
```

```
__mod2__ void InterruptHandler();

main()
{
    _VECTOR(InterruptHandler,14);

    while (1)
    {
        /* Wait here for interrupt to occur */
    }
}

__mod2__ void InterruptHandler()
{
    puts("\n An interrupt has occurred !! \n");
}
```

EVB.h header file

```
/* evb.h - HC11 Registers and interrupt function definitions. */
#undef __WORDreg__
#undef __BYTEreg__
#define __WORDreg__ extern volatile  unsigned short
#define __BYTEreg__ extern volatile  unsigned char
__BYTEreg__ _H11PORTA;               // Port A
__BYTEreg__ _H11PIOC;                // Parallel I/O Control
__BYTEreg__ _H11PORTC;               // Port C
__BYTEreg__ _H11PORTB;               // Port B
__BYTEreg__ _H11PORTCL;              // Alternate Latched Port C
__BYTEreg__ _H11DDRC;                // Data Direction Port C
__BYTEreg__ _H11PORTD;               // Port D
__BYTEreg__ _H11DDRD;                // Data Direction Port D
__BYTEreg__ _H11PORTE;               // Input Port E
__BYTEreg__ _H11CFORC;               // Compare Force
__BYTEreg__ _H11OC1M;                // OC1 Action Mask
__BYTEreg__ _H11OC1D;                // OC1 Action Data
__WORDreg__ _H11TCNT;                // Timer Counter
__WORDreg__ _H11TIC1;                // Input Capture 1
__WORDreg__ _H11TIC2;                // Input Capture 2
__WORDreg__ _H11TIC3;                // Input Capture 3
__WORDreg__ _H11TOC1;                // Output Compare 1
__WORDreg__ _H11TOC2;                // Output Compare 2
__WORDreg__ _H11TOC3;                // Output Compare 3
__WORDreg__ _H11TOC4;                // Output Compare 4
__WORDreg__ _H11TOC5;                // Output Compare 5
__BYTEreg__ _H11TCTL1;               // Timer Control 1
__BYTEreg__ _H11TCTL2;               // Timer Control 2
__BYTEreg__ _H11TMSK1;               // Main Timer Interrupt Mask 1
__BYTEreg__ _H11TFLG1;               // Main Timer Interrupt Flag 1
__BYTEreg__ _H11TMSK2;               // Misc. Timer Interrupt Mask 2
__BYTEreg__ _H11TFLG2;               // Misc. Timer Interrupt Flag 2
__BYTEreg__ _H11PACTL;               // Pulse Accumulator Control
__BYTEreg__ _H11PACNT;               // Pulse Accumulator Count
__BYTEreg__ _H11SPCR;                // SPI Control
__BYTEreg__ _H11SPSR;                // SPI Status
__BYTEreg__ _H11SPDR;                // SPI Data
__BYTEreg__ _H11BAUD;                // SCI Baud Rate Control
__BYTEreg__ _H11SCCR1;               // SCI Control 1
__BYTEreg__ _H11SCCR2;               // SCI Control 2
__BYTEreg__ _H11SCSR;                // SCI Status
__BYTEreg__ _H11SCDR;                // SCI Data
__BYTEreg__ _H11ADCTL;               // A to D Control
__BYTEreg__ _H11ADR1;                // A to D Result 1
__BYTEreg__ _H11ADR2;                // A to D Result 2
__BYTEreg__ _H11ADR3;                // A to D Result 3
__BYTEreg__ _H11ADR4;                // A to D Result 4
__BYTEreg__ _H11OPTION;              // System Configuration Options
__BYTEreg__ _H11COPRST;              // Arm/Reset COP Timer Circutry
__BYTEreg__ _H11PPROG;               // EEPROM Programming Control
__BYTEreg__ _H11HPRIO;               // Highest Priority I-bit and Misc.
__BYTEreg__ _H11INIT;                // RAM/IO Mapping
__BYTEreg__ _H11TEST1;               // Factory Test Control
__BYTEreg__ _H11CONFIG;              // COP, ROM, & EEPROM Enables
/* interrupt function definition */
extern __PASCAL void _ATTACH(void (* func)(void), int vector);
extern __PASCAL void _VECTOR(void (* func)(void), int vector);
```