

Using the C-compiler for Holtek MCUs

D/N : HA0046E

Introduction

Regarding HT-IDE3000 editing, all of the HT-IDE2000 functions are included. However it is important to note that some of the HT-IDE2000 functions, such as peek PX etc, have already been replaced by different style function. If it is necessary to transfer a program then some modification will be necessary.

The objective here is to inform the user, when realising certain functions using mixed language, how to realise them in Holtek C. Also to help the user when using both how to discriminate and about their limitations. For more details consult the HT-IDE3000 User's Guide.

→ Bit Variable Definitions

- Define Bit Variables

bit name

This is the format for the definition in the C language. "bit" defines that the variable type is bit, "name" provides the variable name. When "name" is an overall variable, the system will automatically allocate space.

Example:

```
bit flag1,flag2; //system auto allocates bit space variables
flag1、flag2
main(){
    flag1=flag2=1;
}
```

- define bit variable at indicated address

#define name_ramaddr_bitaddr

The #define instruction can define a bit variable at a RAM location. This is confined to RAMBANK0.

"name" is the bit variable name; "ramaddr" is the RAM address: "bitaddr" is the bit variable address.

Example:

```

#define    ab _40_3    //ab is located at RAM bank0 [40h].3
main() {
    ab=1;
}

```

→ Controlling the I/O ports

The I/O ports can be directly read and written using C. At the beginning of the program, an include statement must be provided for the corresponding MCU as shown below:

```

#include "ht48r70a-1.h"
...
main() {
...
    _pa=_pb=_pc=0;                //clear PA,PB,PC to 0
    _pac=_pbc=_pcc=0;            //setup PA,PB,PC as outputs
...
}

```

→ Table Control

- Construct a Table

const var-type name[num]={date0, data1, data2,...}

The ability to create data tables in an indispensable feature of any programming language, and enables data to be placed into Program Memory spaced and retrieved by the program. In C, const is used to define a constant, at the same time is placed in the Program Memory space. Therefore a constant statement has been defined, which will write a series of indicated values into the Program Memory.

“var-type” is a constant type, and can be an integer or character; “name” is the constant statement name; “num” is the statement size; “data0, data1, data2” is the constant statement contents, and table contents. “data0, data1, data2” can be a byte or a word.

– Example1

```

const    long    array[3]={0x1234,0x5678,0xabcd};
long     temp;
main() {
    temp=array[0];                //temp=0x1234
}

```

– Example2

```

const    char    array[2]={'a', 'b'};
long     temp;
main() {
    temp=array[1];                //temp='a', a character is 61h
}

```

This will generate a table. It is worth noting that each table size must be less than 256 bytes, if the table size exceeds this range, errors will be created.

- Reading the Table contents

When reading the table contents, that is reading the nth data, the data is read beginning with array[0].

Example:

```
const    long array[3]={0x1234,0x5678,0xabcd};
long     t1,t2,t3;
main() {
    t1=array[0];           //t1=0x1234
    t2=array[1];           //t2=0x5678
    t3=array[2];           //t3=0xabcd
}
```

→ **ROMBANK and RAMBANK**

- ROMBANK setup and interrogation

Among the many different Holtek MCU device structures, if the Program Memory capacity exceeds 8K, then it will be subdivided into more than one bank, with each Bank having a capacity of 8K. As an example, taking the HT48RA3, which has three ROMBANKS, with the names BANK0, BANK1 and BANK2. When accessing the ROMBANK, there is a different concept to using either the mixed language or C language program. When assembling it is important to know in which ROMBANK the program is located. For a C language program, it is not necessary to consider this, as the C compiler will automatically setup the ROMBANK and interrogation operations. In this way, in a C program there will be no ROMBANK program statements.

- RAMBANK setup and interrogation

Some MCUs in the Holtek range, such as the HT49 series, which contain an LCD driver, the HTG21 game series, have several RAMBANKs.

Variables can be used to indicate the RAMBANK as in the following definitions:

```
int v1@0x5B; defines v1 to be in RAM bank0 at address 5BH
int v2@0x2F0; defines v2 to be in RAM bank2 at address F0H
```

→ **Example: running light program**

Hardware: Utilises HT48R50A-1 device, PA, PB outputs directly connected to LEDs, system frequency is 4MHz. Lights change every 0.5S.

Program:

```
#include <ht48r50a-1.h>
#pragma vector extern_i @ 0x4           //define interrupt location
#pragma vector timer_i0 @ 0x8
#pragma vector timer_i1 @ 0xc
```

```

#define _te      _0e_4
unsigned int count;
unsigned int light0,light1;
//ISR
void extern_i(){
}

//Timer0 used for 2ms interrupt time
void timer_i0(){
if (count==250)      //0.5sec
{
_c=0;
if (light1==0x1)
{
_c=1;
}
_rrc(&light0);           //running light o/p
_rrc(&light1);
count=0;
}
else count++;
}

void timer_i1(){
}

//MAIN program
void main(){
_clrwdt();
_intc=00;
_tmr0c=0x87;
_tmr0=0xe1;
_pac=_pbc=00;           //setup PA, PB as outputs
light0=0x80;
light1=0x00;
_et0i=1;                //enable interrupt
_emi=1;
_te=1;
while (1)
{
_clrwdt();
_pa=light0;            //PA, PB ouput
_pb=light1;
}
}

```