C Variables

- Declaring a variable as type 'const' will cause the compiler to store the variable in the PIC's program flash memory, not the SRAM.
- Adding the keyword 'volatile' to a variable is a good idea when this variable can be changed or altered outside the sourcefile that declared this variable.

C Functions

- MIOS_*_SRSet and _SRGet Functions refer to the pins in Little-Endian order, so for example:

MIOS_DOUT_SRSet(1, 00000001) Will set the 1st pin (aka Pin 0)…. or
MIOS_DOUT_SRSet(1, 01000000) Will set the 7th pin (aka Pin 6)

C Optimizations

- How to mix C and ASM

SDCC Bugs/Workarounds

Some of these bugs have first been described in a german thread in the forum.

Array Access

Sometimes the transfer of an array between modules does not work properly, e.g. file 1:

```
unsigned char MIDIValues[8];
```

file 2:

```
MIOS_MIDI_TxBufferPut(MIDIValues[1]);
```

Instead, you need to do something like

```
unsigend char value = MIDIValues[1]; //explicite temp variable
MIOS_MIDI_TxBufferPut(value);
```

Large Arrays

Arrays with more than 256 elements will produce compile (in fact linker) errors:

```
unsigned char myArray[256]; // will work
unsigned char myArray[257]; // will not be linked!

unsigned char myArray[64][4]; // will work
unsigned char myArray[64][5]; // will not be linked!
```

Thanks to Thomas for testing some workarounds with multiple single-dimensional arrays.

Parenthesis

Always use parenthesis around expressions like

```
myarray[a+b];
```

instead use

```
myarray[(a+b)];
```

Preprocessor #ifs

Avoid #ifdef and #if preprocessor-statements wrapped around declarations and function prototypes. Even if the preprocessor's #if statement is true (eg defined as '1'), any access to it's vars and functions from outside these wrapped statements produce a compile-warning:

```
#define TEST 1

#if TEST
  unsigned char testvar;
#endif /* TEST */
```

```
void testfunction(void) {
  unsigned char c = testvar + 1;  // access to testvar produces compiler
error!
}
```

Zero Compare

Avoid comparisons of `unsigned char` with `0`, e.g.

```
unsigned char i;
   for (i = 0; i < 0; i+ü) {
      //body
   }
```

`0` could be a constant that was defined using `#define`, e.g. the number of motorized faders. But you have no motorized faders… The main problem consists in the fact that your code depends on what else is done around the comparison or in the body. This provokes completely erratic behaviour.