

```

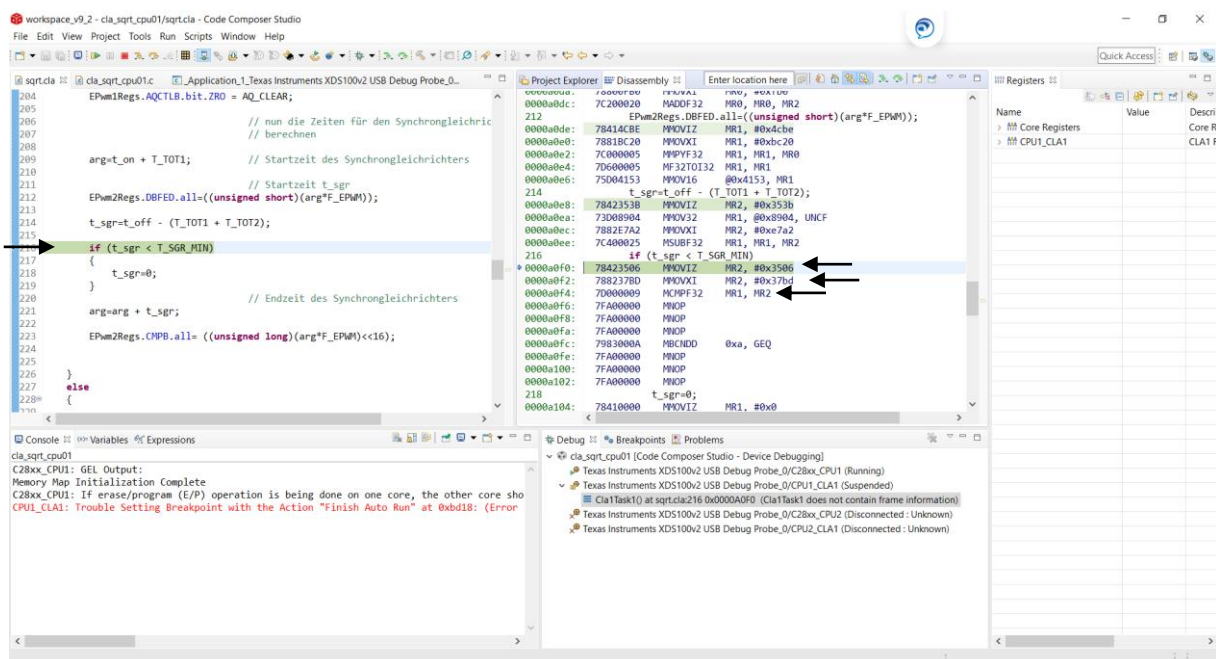
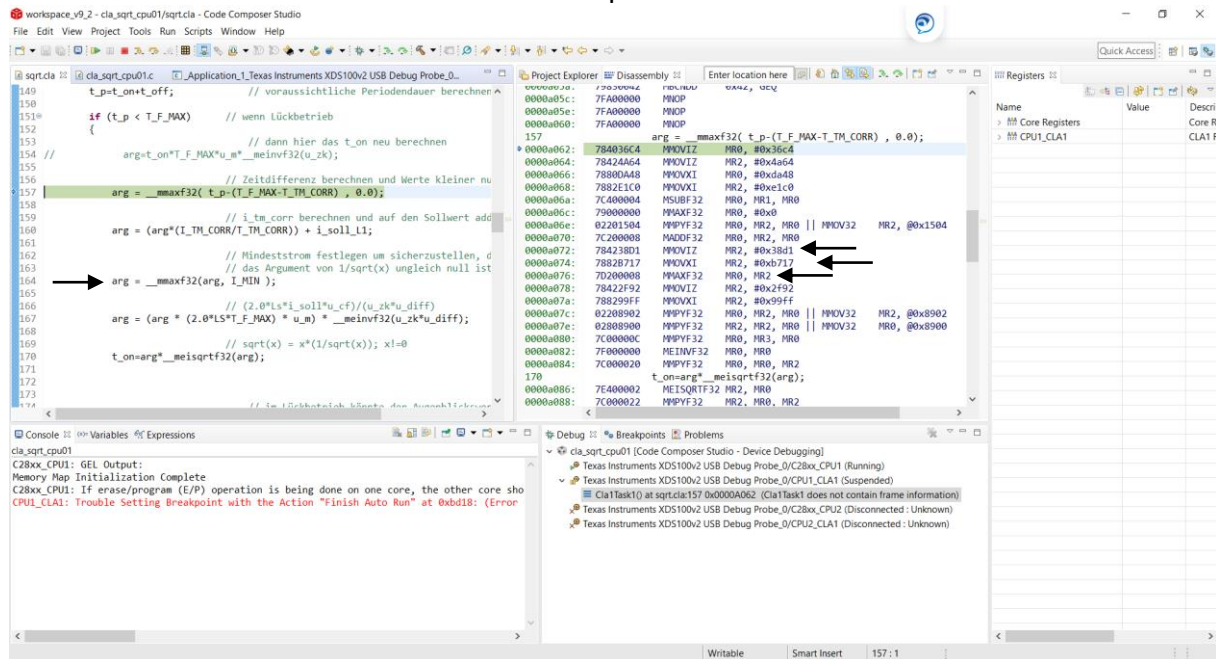
#define T_SGR_MIN    (500e-9)           // full precision
#define T_SGR_MIN_R  (499.188899e-9)    // reduced precision, calculated with float/hex Tool by myself

#define I_MIN        (0.1e-3)           // full precision
#define I_MIN_R      (99.658966064453125e-6) // reduced precision, calculated with float/hex Tool by myself

```

I have defined two constants in my code with full and reduced precision. The values with the reduced precision should be calculated later with the preprocessor. The values with the reduced precision yields to a more efficient code, because the lower 16bit of the floating point value are zero and the compiler is able to use the mnemonics with the integrated #16FHi constant.

Code with the full precision constants



Code with the reduced precision constants

The screenshot shows the Code Composer Studio interface with the assembly view of the `sqrt.cla` file. The assembly code is for the `sqrt.cla` task, and the assembly view shows the instructions for the `sqrt.cla` task. The assembly code is for the `sqrt.cla` task, and the assembly view shows the instructions for the `sqrt.cla` task.

The screenshot shows the Code Composer Studio interface with the assembly view of the `sqrt.cla` file. The assembly code is for the `sqrt.cla` task, and the assembly view shows the instructions for the `sqrt.cla` task. The assembly code is for the `sqrt.cla` task, and the assembly view shows the instructions for the `sqrt.cla` task.

With the reduced precision only one mnemonic is needed to perform the operation.