Displaying version information

Whilst it is possible to use the generated version string from **getVersion** on its own, for example

```
#include "_version.h"

void printVersion() {
    fputs("Current version " GIT_VERSION "\n");
}
```

I personally augment the auto generated information with additional static fields and have written some C code **_version.c** that shows this information and a also a Windows resource file **_version.rc** that populates the file properties.

Supporting this code are two include files

```
_version.h      as generated by getVersion
appinfo.h       containing addition static information
```

The static information takes the form or #defines for C/C++ and the current set is as shown below, note only the APP_NAME is required.

#define	Use
APP_NAME	Name of the application
APP_PRODUCT	Primarily for ported applications, this the formal product name. Can also be used to reflect major structural changes to application. It defaults to APP_NAME
APP_OWNER	This is the copyright owner of the original application. Defaults to my name
APP_MODS	A string used a add supporting information, about modifications made. For example ported code, could use the string "port Mark Ogden". if present, it is displayed inside < >.-
APP_DESCRIPTION	A brief description of the application - only shown for full version info
APP_CONTRIBUTOR	A list of contributors - only shown for full version info
ALL_EMAIL	A support email address - only shown for full version info

Although the code to display the version information includes appinfo.h, for groups of applications it may make sense for appinfo.h, to include app specific #defines and to provide default values for other items.

For example, as part of the work I have done on reverse engineering of the Hi-Tech C complier, I set up appinfo.h as

```
// static version information
// overrides are in _appinfo.h in the source directory
```

```
#ifndef _APPINFO_H_
#define _APPINFO_H_
#include <_appinfo.h>       // appication specific defines
#ifndef APP_PRODUCT
#define APP_PRODUCT APP_NAME "-3.09"
#endif
#ifndef APP_OWNER
#define APP_OWNER       "HI-TECH"
#endif
#ifndef APP_MODS
#define APP_MODS        "port Mark Ogden"
#endif
#ifndef APP_DESCRIPTION
#define APP_DESCRIPTION APP_PRODUCT " reverse engineered to modern C"
#endif
#define APP_EMAIL       "support@mark-ogden.uk"
#endif
```

In the case of zas, I then setup _appinfo.h as

```
#define APP_NAME    "zas"
#define APP_CONTRIBUTOR "Andrey Nikitin"
```

showVersion

In the C code there is a single function, with prototype

`void showVersion(bool full);`

Which is defined in showVersion.h along with a C macro to standardise showing version information

`CHK_SHOW_VERSION(argc, argv)`

that takes main's argc and argv values and checks for a single -v or -V command line argument and calls **showVersion** with full set to true if it was -V and exits. Typical usage would be

```
int main(int argc, char **argv) {
    /* optional declarations */
    CHK_SHOW_VERSION(argc, argv);
    /* rest of main's code */
}
```

showVersion, takes the APP__* and GIT_VERSION information and displays the following, with [xxx] information being optional

```
APP_PRODUCT " (C) " APP_OWNER ['<' APP_MODS '>'] '  ' GIT_VERSION

For full version information, the following extra information is shown

[APP_DESCRIPTION]
[Contributors: APP_CONTRIBUTOR]
architecture " build: " build date
["Support email: " APP_EMAIL]

Where architecture reflects code target 32/64 bit and whether it is a debug
version.
For MSVC and GCC the compiler version is also shown
```

Using the zas example, the simple version information shown is

```
zas-3.09 (C) HI-TECH <port Mark Ogden> 2023.4.23.5
```

and for the full version information

```
zas-3.09 (C) HI-TECH <port Mark Ogden> 2023.4.23.5
zas-3.09 reverse engineered to modern C
Contributors: Andrey Nikitin
64 bit debug build: (msvc 19.42.34435) Dec 10 2024 11:31:33
Support email: support@mark-ogden.uk
```

```
Updated by Mark Ogden 10-Dec-2024
```