MicroSys

Software Getting Started

Software enablement guide SBC-S32G

V 0.3

Table of Contents

1 General Notes

1.1 Warranty

To the extent permissible by applicable law all information in this document is provided without warranty of any kind, whether expressed or implied, including but not limited to any implied warranty of satisfactory quality or fitness for a particular purpose, or of non-infringement of any third party's rights. We try to keep this document accurate and up-to-date but we do not make any warranty or representation about such matters. In particular we assume no liability or responsibility for any errors or omissions in this document.

MicroSys Electronics GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product.

MicroSys Electronics GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

1.2 Links

We make no warranty about any other sites that are linked to or from this document, whether we authorise such links or not.

1.3 Liability

To the extent permissible by applicable law, in no circumstance, including (but not limited to) negligence, shall we be liable for your reliance on any information in this document, nor shall we be liable for any direct, incidental, special, consequential, indirect or punitive damages nor any loss of profit that result from the use of, or the inability to use, this document or any material on any site linked to this document even if we have been advised of the possibility of such damage. In no event shall our liability to you for all damages, losses and causes of action whatsoever, whether in contract, tort (including but not limited to negligence) or otherwise exceed the amount, if any, paid by you to us for gaining access to this document.

MicroSys Electronics GmbH
Muehlweg 1
82054 Sauerlach
Germany

Phone: +49 8104 801-0
Fax: +49 8104 801-110

1.4 Offer to Provide Source Code of Certain Software

This product contains copyrighted software that is licensed under the General Public License (“GPL”) and under the Lesser General Public License Version (“LGPL”). The GPL and LGPL licensed code in this product is distributed without any warranty. Copies of these licenses are included in this product.

You may obtain the complete corresponding source code (as defined in the GPL) for the GPL Software, and/or the complete corresponding source code of the LGPL Software (with the complete machine-readable “work that uses the Library”) for a period of three years after our last shipment of the product including the GPL Software and/or LGPL Software, which will be no earlier than December 1, 2010, for the cost of reproduction and shipment, which is dependent on the preferred carrier and the location where you want to have it shipped to, by sending a request to:

MicroSys Electronics GmbH
Muehlweg 1
82054 Sauerlach
Germany

In your request please provide the product name and version for which you wish to obtain the corresponding source code and your contact details so that we can coordinate the terms and cost of shipment with you.

The source code will be distributed WITHOUT ANY WARRANTY and licensed under the same license as the corresponding binary/object code.

This offer is valid to anyone in receipt of this information.

MicroSys Electronics GmbH is eager to duly provide complete source code as required under various Free Open Source Software licenses. If however you encounter any problems in obtaining the full corresponding source code we would be much obliged if you give us a notification to the email address gpl@microsys.de, stating the product and describing the problem (please do NOT send large attachments such as source code archives etc to this email address)

2 Introduction

Thank you for choosing the MicroSys SBC-S32GXXX Single Board Computer system. This manual should help you gain a good understanding of the software and how to enable you to use it.

2.1 Short Description

Any “SBC” labeled system is a small computer system consisting of

- the MPX-S32GXXX module, based on NXP’s S32G Vehicle Network Processor
- and the CRX-S32G carrier board.

The MPX, also short called module, can be exchanged.

2.2 Terminology

For better understanding, generic terms are used across this documentation.

The “MPX-S32GXXX module” is referred as module.

The “CRX-S32G” is referred as carrier.

Software that might have restricted access and is therefore optional is called “restricted firmware”.

Whenever the term “BSP” is used, it refers to the delivery containing the original NXP BSP, all open-source projects linked to it, patches from MicroSys and potentially restricted firmware. In case the author talks about the unchanged NXP delivery, it is called “NXP BSP”.

3 Build setup

All MicroSys releases are built on top of the public NXP BSP releases.

3.1 NXP firmware binaries

NXP provides firmware for peripherals like PFE, HSE and LLCE. The BSP release provides infrastructure to integrate those as a firmware repository. MicroSys doesn't use the externalsrc providing structure NXP does. This can help you for automation environments.

In case you want to use this as well, you need to get them from NXP directly by accepting their License agreement. If you have any trouble getting in touch with NXP colleagues, please contact us, we can arrange this.

3.1.1 Firmware repo structure

The repository shall use the following structure:

- pfe
  - s32g2
    - s32g_pfe_class.fw
    - ….
- hse
  - s32g2
    - HSE_FW_S32G2XX_0_1_0_14
      - ….
- llce
  - s32g2
    - dte.bin

The structure above is just a short, incomplete form of this repo of a specific version.
You can also use just a subset of them.

The repo assignment is controlled within "meta-microsys-auto/conf/microsys-distro-common.conf". There are samples to provide the source as git repo. You can also use other formats like a zip file with the same structure. Be careful about relative paths for different archive types.

You can override or patch those values for your use-case.

3.2 Build environment

The build works with the same settings as normal NXP BSP builds work as well. Please refer to their documentation for this.

Extract the meta-microsys-auto directory into the source repo. Usually at the same level as meta-alb.

Patch the meta-alb and poky with files provided in meta-microsys-auto/patches/meta-alb

3.2.1 Sample docker image

There is a sample docker image available which is suitable as container to build the BSP. It handles the pitfalls as well as patches we require.

The docker setup is in an alpha state and has some open points. These may not be problematic for you but consider them.
In case you use your own environment, check the section about patches in the dockerfile and adapt those. Doing those patches is mandatory.

Make sure you have docker installed

Make sure you placed the downloaded files mentioned in the “TODO” section of the Dockerfile in the same directory as “Dockerfile”. Namely microsys metalayer and optionally your NXP Firmware

Do to the directory with the Dockerfile and build the container:

```
docker build -t nxpbuildcontainer:bsp42 ./
```

Create directories for sstate-cache, downloads and deploy
Run the container:

```
docker run --rm -it -v <Storage
directory>/sstate/:/home/dev/fsl-auto-yocto-bsp/sstate-
cache -v <Storage directory>/downloads/:/home/dev/fsl-auto-
yocto-bsp/downloads -v <Storage
directory>/deploy/:/home/dev/deploy nxpbuildcontainer:bsp42
bash”
```

Run

```
source nxp-setup-alb.sh -m s32g399ar5sbc3 -j 8 -t 12
```

Replace the machine identifier with the one you want to build.
Replace the -t 12 with the number of cores your machine has to speed up builds.
Reduce the -t 12 in case you have not much RAM. Minimum RAM should be the double of -t parameter to be on the safe side.

To build, run

```
bitbake microsys-image-auto
```

The ready to use images are located at in

```
/home/dev/fsl-auto-yocto-
bsp/build_s32g274ar2sbc3/tmp/deploy/images/…
```

Copy all files you need to /home/dev/deploy. These will appear in the mapped directory of the docker run. Otherwise, those files will all be deleted once you exit the container.

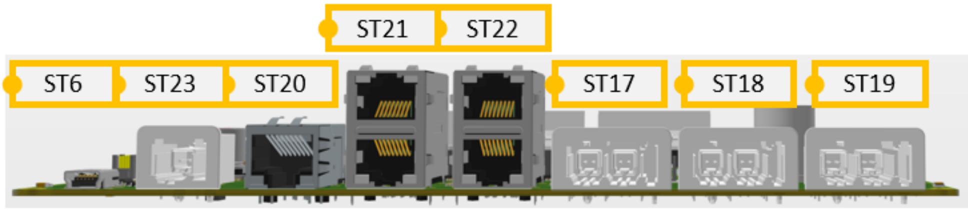
You can rebuild the container/map the metalayers for rebuilding and more advanced development. If you map downloads and sstate-cache directory, the build is done within a couple minutes. A clean building takes an hour on high performance systems.

3.2.2 Known issues

- The inclusion of NXP firmware seems to be not working. They are kept unbuilt. This is under investigation. You can still build without the firmware images/install them in the final image manually.
- The build stores the tmp directory in the docker root fs.
  This means you need ~100GB of space there or you set in your docker daemon config a proper "data-root" where you have lots of space.
- Github sources fetching might be interrupted. In case of fetch errors, just restart the build. It will save all successful downloads in the mapped directory. Don't clean that directory, so future builds will be more stable.

4 Peripheral related

4.1 Ethernet



The carrier has a variety of Ethernet connectors. Please refer to the hardware manual for internal connection of those. This document will only mention internal connections and assignments if required for the software view.

	Name in Uboot	Name in Linux
ST23	N/A	**pfe0 (1GBit)**
ST20	N/A	**pfe0 (100Mbit)**
ST21A (top)	**ethernet (1Gbit)**	**eth0 (1Gbit)**
ST21B (bottom)	**pfe0 (1Gbit)**	**pfe0 (1Gbit)**
ST22A (top)	**pfe1 (1Gbit)**	**pfe1 (1Gbit)**
ST22B (bottom)	**pfe2 (1Gbit)**	**pfe2 (1Gbit)**
ST17A (left)	N/A	**pfe0 (100Mbit)**
ST17B (right)	N/A	**pfe0 (100Mbit)**
ST18A (left)	N/A	**pfe0 (100Mbit)**
ST18B (right)	N/A	**pfe0 (100Mbit)**
ST19B (left)	N/A	**pfe0 (100Mbit)**
ST19 (right)	N/A	**pfe0 (100Mbit)**

Be aware that pfe0 is always set to “link up” in Linux, because it is going through the SJA Switch that is providing a link. That’s the same reason the “pfe0” appears multiple times on different ports.

For this reason, pfe0 will not acquire an IP on connection of ethernet via DHCP if the connection was established after booting. Run

```
udhcpc -i pfe0
```

In uboot you can switch between pfe1 and pfe2 by “env set ethact eth[2|3]”. Those mappings are also printed on uboot boot console.
As pfe0 is going through the SJA1110 which is not yet initialized, it’s not available in Uboot.

4.1.1 2.5 Gbit support

The board supports 2.5 Gbit for the pfe0 connection. It is the connection bandwidth between the processor and the SJA switch. The 2.5 Gbit is therefore an internal connection and not accessible outside of the chip. Enabling that reduces risk of a bottleneck at the uplink.

2.5Gbit is not supported by default in BSP42. This is an active topic on our side.

4.2 SJA-Firmware

If your network supports the IP range 192.168.0.0/24, you can see statistics about the SJA1110 under http://192.168.0.165/uplink.html

Those indicate which ports are active.

4.3 Storage eMMC/SD card

eMMC and SD card are not supported simultaneously. There is a multiplexer controlled by a microcontroller (since module Rev 5, before a EDIP switch) to select one.

You can see which one is selected by

```
mmc info
```

command in uboot. You get “MMC version 5.X” for eMMC and “SD version: X.Y” dependent which one is executed.

Module Rev5 and later, you can switch temporarily to eMMC by the command

```
i2c dev 0;i2c mw 0x10 0x12 <VALUE>
```

VALUE == 1 means eMMC, 0 is SD card. Run

```
mmc rescan
```

after a switch. This way you can load data from SD, do the switch, and flash data to eMMC. This is a quick and easy way to flash eMMC. Especially if there is no direct network connection.

4.4 Storage QSPI

The QSPI, better called NOR flash on the module provides reliable storage over long time. It can be flashed from internal or external tooling.

Easiest way is using the uboot in SD card mode.

```
i2c dev 0;i2c mw 0x10 0x12 <VALUE>
```

4.5 Storage EEPROM

There is an EEPROM on the module which holds the RCW defining the boot mode of S32G. It is also the default location for the u-boot environment.

On the default module, there are two EEPROMs (0x50 and 0x56) which can be flipped by a dipswitch.

Be careful writing to those as this might make the board unbootable.

Some changes on the EEPROM are only effective after a power cycle.

4.6 Serdes config

The serdes is the underlying system for PCIe and some Ethernet connectors.

It needs to be configured at uboot. In case this is done wrong, it has an effect on the devicetree the Linux kernel will be launched with.

The board natively supports the following modes:

- M2 slot active + 1Gbit on pfe0
- M2 slot inactive + 2.5Gbit on pfe0

You can alter the modes with the command “serdes mode <m2|2G5>” in uboot. Changing that will alter the EEPROM contents. Replacing/reflashing the SD card/eMMC/QSPI doesn’t restore the settings.

Depending on this mode, the uboot and Linux will select at runtime which serdes config (part of “hwconfig” of uboot), SJA Firmware and network config shall be used.

In case you persist, an environment variable called “hwconfig_forced”, this will override the precompiled two configs for M2 on/off as mentioned above.

For special operations, you can also set the serdes frequency manually without mode guidance. See the help of the “serdes” command for further details.

4.6.1 Custom Serdes config

The Microsys Uboot supports setting special serdes hwconfig parameters in case you use a custom carrier. It overwrites the M2/2G5 settings.
Get in touch with us to provide further details.

5 Boot config

5.1 HSE usage

In case the HSE is configured, it is launched before the A53 or M7 application cores are used.

Information about how to configure HSE is found in Build setup/NXP firmware binaries.

5.1.1 Known pitfalls

In case the launching of the HSE fails, the boot will not continue. As HSE doesn't print anything on serial, there is no reaction from the board. Depending on the other configuration, a watchdog might trigger resets.

HSE has different versions which are not compatible between CPU SoC revisions. Always use the intended versions.

5.2 A53 boot

By default, the MicroSys metalayer configures the booting of A53 core 0 directly. There is no lockstep configured.
BootROM loads the initial image (BL2) to SRAM and launches it. In default configuration, BL31, UBoot and Linux are booted.

5.3 M7 boot

In case one activates the M7 boot by passing "m7boot" as distro feature by putting

DISTRO_FEATURES_DEFAULT:append = " m7boot "
require conf/machine/include/m7-boot.inc
ATF_IMAGE_FILE = "fip.s32-qspi.m7"
ATF_IMAGE = "arm-trusted-firmware"

in the microsys-auto.conf/your conf file, it will include an M7 application.

The building produces one firmware image without the m7 and one containing it. The SDCard images contain the M7 firmware. They can be observed by the ".m7" suffix.

BootROM will load the A53 plus M7 image to SRAM and execute the M7 on the M7_0 core.

The sample application provided is initializing the A53_0 core and launching the already loaded BL2 on it. After this, it puts the M7 core to sleep by exiting itself. Despite the default M7 boot config of NXP, MicroSys is not initializing the A53 in lockstep mode.

Due to absence of UART drivers, clock initialization and more, the M7 application cannot print anything. One can verify by debugger that M7_0 is now in the end label of the m7 application.

You can also see the data of the M7 application with a debugger. With BSP42 G3 builds you can see the code like here. One could use the M7 for further computation or any other need.

5.3.1 Known pitfalls

As of BSP42, the M7 image is prepended to the BL2 image.
In case one puts more code into the M7 image, SRAM is required to hold it.
Consider moving the BL2 load address to another location to make space for it.

The current implementation of BSP42 relies on concatenation of M7 and A53 images. This means that M7 is loaded in front of the A53 application. This might limit more complex solutions. After BL2 has jumped to BL31, its memory location can be reused. Consider synchronizing this event in M7 or treat the BL2 memory space as reserved the whole lifetime.

6 History

Date	Version	Change Description
2024-11-21	**0.1**	Initial version
2024-12-06	**0.2**	Added Docker build description and Bootmodes
2024-12-09	**0.3**	Adapt and test docker guide for BSP42

Table 6-1 Document History