ds30 Loader
Main manual

DigSol Sweden AB

Table of contents

Introduction

ds30 Loader

ds30 Loader is a boot loader supporting PIC12, PIC16, PIC18, PIC24, dsPIC, and PIC32 families of MCUs from Microchip. It supports all devices in each family out of the box (those in production). The firmware is written in assembler. The PC clients run on Windows, Linux, and Mac OS X.

Prerequisites and Requirements

Requirements for the different parts of ds30 Loader is found in their respective manual.

Trademarks

Why use a boot loader

A boot loader usually allows software upgrade with cheap or generally available equipment such as an RS232 port, as opposed to specialized and expensive equipment such as a PIC programmer. Write time might also be lower with a boot loader.

Drawbacks of using a boot loader include added boot-up time and increased flash memory usage.

Usage

Operating system considerations

Windows

N/A

Linux

The user running ds30 Loader should be a member of the dialout group. To add a user in the dialout group run the following command as a super user:

```
useradd –G dialout username
```

Mac OS X

The user running ds30 Loader should have sufficient rights to read from and write to the desired port.

Install the required tool suite

The required compiler/assembler tool suite needs to be installed before the boot loader firmware is opened in MPLAB IDE. If the tool suite is not installed when the project is opened important settings may be lost. See the firmware manual for information about the required tool suite(s).

Modify firmware settings

Refer to the firmware manual for detailed information.

- Start Microchip MPLAB IDE.
- Open the firmware by clicking Open on the project menu, and then browse to the ds30loader.mcp file which is located in the firmware directory.
- Open settings.inc(assembler firmware) or board_xxx.h(C firmware) by double-clicking it in the project tree to the left. Review all settings and modify when needed.
- Open user_code.inc(assembler firmware) or board_xxx.c(C firmware) by double-clicking it in the project tree to the left. Review the available macros/functions and add init/exit code if needed.
- Use exactly the same configuration bits from the application in the boot loader unless you have a good reason not to.

Build the firmware

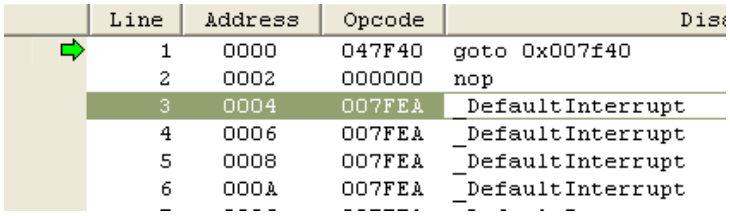
Build the firmware by clicking Build all on the project menu. Notice any warnings in the output window. Correct any errors that appear in the output window. Refer to the firmware manual for detailed information about ds30 Loader specific warnings and errors.

Write firmware

Write the firmware to your device using your favorite programmer. For detailed information, refer to the firmware manual.

Prepare the application

You need to make sure there is a goto placed at location 0x00 in your application. In most cases your linker already does this for you. To check if it does you can pick Progam Memory from the View-menu in MPLAB. It will look something like the screen-shot below. The GUI will let you know if it does not find a goto.



Line	Address	Opcode	Dis
1	0000	047F40	goto 0x007f40
2	0002	000000	nop
3	0004	007FEA	_DefaultInterrupt
4	0006	007FEA	_DefaultInterrupt
5	0008	007FEA	_DefaultInterrupt
6	000A	007FEA	_DefaultInterrupt

Setup the host application

- Start the GUI.
- Select the desired communication settings and pick your hex-file. For details, refer to the GUI manual.

Write the application

There are three ways to invoke the boot loader and perform the write.

Procedure 1

1. Press the write button in the GUI.
2. Reset the PIC/dsPIC.

Procedure 2

1. Reset the PIC/dsPIC.
2. Press the write button in the GUI before the boot loader times out.

This procedure may be more reliable than procedure 1.

Procedure 3

0. Implement a reset command in the application. See the reset part in the configuration section in this document for more information.
1. Configure the reset command under the reset the in the GUI. Advanced mode need to be enabled first.
2. Press the write button in the GUI.

Wait for the write operation to complete

Firmware matrix

The following firmwares are available. For detailed information refer to the firmware manual.

	UART	Sw UART*	CAN*	I²C*	SD card*
PIC12F	X	*	n/a	see ds30 HEX Loader	see ds30 SD card loader
PIC16F	X	*	n/a		
PIC18F	X	*	*		
PIC24FK	X	*	n/a		
PIC24FJ	X	*	n/a		
PIC24H	X	*	*		
PIC24E	*	*	*		
dsPIC30F	X	*	*		
dsPIC33F	X	*	*		
dsPIC33E	*	*	*		
PIC32MX	*	-	*		
PIC32MZ	coming	coming	coming		

* Only available in the commercial version

Clients

ds30 Loader Console

A command line interface that runs on Windows, Linux, and macOS. Refer to the console manual for more information. Require .NET or Mono.

ds30 Loader GUI

It is a graphical user interface that runs on Windows, Linux, and macOS. Refer to the GUI manual for more information. Require .NET or Mono.

ds30 Loader .NET API

The API is used to integrate boot loader functionality in a .NET application. It is ordered separately; contact DigSol Sweden for more information. Require .NET or Mono.

ds30 Loader Native console

A command line interface that does not require any framework to be installed. Supports Windows, Linux and macOS

Feature matrix

	.NET applications	Native Console
UART	x	x
CAN	See the CAN section of this document	All adapters supporting SocketCAN
Write flash	x	x
Write EEPROM	x	x
Write config bits	x	
Automatic baud rate detection	x	
Echo verification	x	
Configurable timings	x	x
Reset by command	x	x
Reset by rts	x	
Read flash	x	
Read EEPROM	x	

CAN

The ds30 Loader supports boot loading on the CAN bus. See table 1 for available firmwares and table 2 for supported CAN adapters. CAN operation is not available in the free edition.

Device Family	Firmware available	Supported DLC
PIC12	n/a	-
PIC16	n/a	-
PIC18	yes	1-8*
PIC24F	n/a	-
PIC24FJ	n/a	-
PIC24H	yes	1-8*
PIC24E	yes	1-8
dsPIC30F	yes	1-8
dsPIC33FJ	yes	1-8*
dsPIC33E	yes	1-8
PIC32	yes	1-8

Table 1. Available CAN firmwares, * = older version may only support a DLC of 1.

Adapter	Supported bit rates	Windows		Linux		Mac OS X	
		x86	x64	x86	x64	x86	x64
IXXAT	10k, 20k, 50k, 100k, 125k, 250k, 500k, 800k, 1M	X	X	-	-	-	-
Kvaser	50k, 62k, 83k, 100k, 125k, 250k, 500k, 1M	X	X	-	-	-	-
Vector	10k, 20k, 50k, 100k, 125k, 250k, 500k, 800k 1M	X	X	-	-	-	-
Peak	5k, 10k, 20k, 50k, 100k, 125k, 250k, 500k, 1M	X	X	-	-	-	-
Lawicel USBtin	10k, 20k, 50k, 100k, 125k, 250k, 500k, 800k, 1M	X	X	X	X	X	X
SYS TEC	10k, 20k, 50k, 100k, 125k, 250k, 500k, 800k, 1M	X	X				
IS-CAN	10k, 20k, 50k, 100k, 125k, 250k, 500k, 800k and 1M	X	X	-	-	-	-
ESD	Contact						
IntrepidCS							
Komodo							

NI-CAN	-
X = supported - = not supported	

Table 2. Supported CAN adapters in ds30 Loader .NET applications.

Lawicel

There may be issues related to Lawicel CAN adapters, see the Trouble shooting-Known issues section in this document.

The Lawicel CAN232 adapter is shipped with the baud rate set to 57600. The baud rate can be increased for a performance improvement; it is done in the configuration window available in ds30 Loader GUI. The serial interface baud rate cannot be changed with the ds30 Loader console application.

If the ds30 Loader console application is used, the serial interface baud rate must be set manually in the Lawicel port configuration file to reflect the baud rate the Lawicel adapter is configured for.

CAN232

When auto poll is disabled the GUI terminal will not receive data. Transmit still works.

IXXAT

There may be issues related to IXXAT CAN adapters, see the Trouble shooting-Known issues section in this document.

The bitrates can be fully customized by editing the ixxatBitRates.xml file. It is located in the same directory as the ds30 Loader executables.

Vector

The bitrates can be fully customized by editing the vectorBitRates.xml file. It is located in the same directory as the ds30 Loader executables.

Configuration

This section is valid for ds30 Loader GUI, ds30 Loader console and the ds30 Loader API.

Basic settings

Port name

The name of the port that will be used to communicate with the boot loader firmware.

Baud rate

The baud/bit rate that the port will be setup for prior to communicating with the boot loader firmware.

Write flash

When checked, the flash memory is written during a write operation. This is disabled if the chosen hex-file does not contain any program memory locations.

Write eeprom

When checked, the eeprom memory is written during a write operation. This is disabled if the chosen hex-file does not contain any eeprom memory locations.

Read flash

When checked, the flash memory is read during a read operation.

Read eeprom

When checked, the eeprom memory is read during a read operation.

Serial

Flow control

Sets the serial port flow control mode.

Parity

Sets the serial port parity mode.

Stop bits

One or two stop bits can be selected.

CAN

PIC ID

This setting is only available for CAN ports, it specifies the PIC CAN ID, and it should be set to the same value as in the boot loader firmware. This can be entered as decimal (example 10) or hex (example 0xa).

GUI ID

This setting is only available for CAN ports. It specifies GUI CAN ID and it should be set to the same value as in the boot loader firmware. This can be entered as decimal (example 10) or hex (example 0xa).

Extended frames

When set extended frames are sent and only extended frames are received.

DLC

This setting determines how many bytes of data will be send in each CAN frame. Note that not all CAN firmwares support high DLC than 1.

TCP socket

Address

The address and port nr to connect to. The format is address:portnr. The address can be either an ip addres or a name which is resolved when the connection is established.

Advanced

This page contains advanced settings that are normally not used.

Don't relocate application reset vector

When enabled, the first instructions in the hex file will not be moved to just before the boot loader. This can be useful when writing part of an application or only data to flash.

Don’t point reset vector to boot loader

Usually a goto to the boot loader is placed at 0x00. When this option is checked, the first few words at 0x00 will not be modified. If a hex file that contains data in the first page is written with this option enabled and the boot loader firmware does not feature goto protection, the boot loader will not work any more because it will not be called on start-up.

Don’t write empty pages

When this is enabled pages that only contain 0xff will not be written. This may decrease write time and may also resolve the “The hex-file contains code that will overwrite the boot loader” message.

Custom boot loader

Allows proper operation with custom boot loader firmwares where size and placement has changed. Great care must be taken to set correct custom placement and size, they must match the BLPLP and BLSIZEP defines in the firmware. If a write is made with the wrong custom parameters, the boot loader may get broken if the firmware doesn’t feature overwrite protection.

Allow overwrite of the boot loader

When a hex file is parsed there is control to make sure that the contents will not overwrite the boot loader, by enabling this setting that control is disabled. This does not bypass the firmwares own overwrite protection if any.

Put checksum before application vector (experimental, not supported)

Calculates a 16-bit (14-bit for PIC12 and PIC16) checksum value of the program memory and stores it just before the user application goto that in turn is placed just before the boot loader.

Write configuration bits

When checked, the configuration bits is written during a write operation. This is disabled if the chosen hex-file does not contain any configurations bits or if the hex-file contains more configuration bit locations than the selected device has. Writing configuration bits are only possible once per power on cycle.

One-way communications

The host application will insert a delay after sending a command instead of waiting for a response from the firmware.

Auto baud rate

When enabled, an auto baud rate synchronization character is sent prior to the write, read and check for boot loader operations. The firmware must be set-up for auto baud rate detection. Not all firmwares come with this feature.

Echo verification

If the device hardware or firmware is set-up to echo all received data this option must be enabled. When enabled, the ds30 Loader engine will compare all sent data with the echo and give an error if there is a mismatch.

Timing

Hello timeout

Specifies the time to send the hello command before giving up when no response is received. If set to 0 there is not timeout.

Poll time

Specifies the interval at which the hello command is sent to the boot loader.

Timeout

Specifies the communication timeout.

Delay after port open

This delay is issued right after the port is opened. It can usually be kept at 0.

Reset

RTS

When enabled, the RTS pin is held high prior to communication with the firmware. The value of reset time specifies the time the RTS pin is held high. This is only available for serial ports and hardware support must be present.

DTR

When enabled, the DTR pin is held high prior to communication with the firmware. The value of reset time specifies the time the DTR pin is held high. Hardware support must be present. This is only available for serial ports.

Command

When enabled, a command is sent to the device to request reset or loading of the boot loader. When a CAN port is used, the macros $PICID and $GUIID can be used, they will be expanded to a two or four-byte big endian value depending on if extended is used or not.

The reset time specifies the time to wait after the command is sent before trying to communicate with the boot loader.

The format of the data is hexval1;'string';"string"....
Example hex only: 0;11;f;ab;3e
Example strings only: 'Hello';"Reset"
Example mixed: a5;'Reset'

ID

The CAN id to be used when sending the reset command.

DLC

The CAN DLC to use when sending the reset command.

Baud rate

The baud rate to use when sending the reset command.

Activation

RTS

When checked, the RTS pin is used to activate the device by holding it high during a write operation. Hardware support must be present.

DTR

When checked, the DTR pin is used to activate the device by holding it high during a write operation. Hardware support must be present.

Security

Password

The password will be sent before any boot loader operation is performed.

Operation

Parsing of hex-file

The following operations take place during the parsing of a hex file:

Validation

The hex file is validated. Line checksums, file format and file completeness is checked. If any error is detected, the parsing is aborted.

Data collection

All data found in the hex-file that fits in the selected devices memory area are stored in RAM buffers.

Check of data that could overwrite the boot loader

If data found in the hex-file belong to the same memory space as the boot loader, a warning is displayed and the Write program check box gets disabled.

Check and move of GOTO

This step is not performed if the hex-file is encrypted. If no GOTO is found at address 0x00 the boot loader does not know how to load the user application and the Write program check box gets disabled. If a GOTO at 0x00 is found, it is moved to the two words just before the boot loader.

Counting of data

The data in the buffers are counted for presentation.

Set GOTO to boot loader

A GOTO to the boot loader is inserted in the first words beginning at 0x00.

Write

The following operations take place during a write operation:

Raise of process priority

If the poll time is set lower than 100ms, the GUI process priority is raised to above normal.

Device reset

If activated in the reset tab, the device is reset.

Auto baud rate

If auto baud rate is checked, the auto baud rate synchronization character (0x55) is sent using poll time until time out is reached or the boot loader responds.

Find boot loader

The hello command is sent using poll time until time out is reached or the boot loader responds. The boot loader responds with device id and firmware version. The received device id is checked against the selected device in the GUI.

Determine boot loader size

The boot loader size is determined based on the firmware version.

Parse hex-file

The hex-file is parsed if needed depending on if the determined boot loader size is different from the one used during last parse.

Write

Program, eeprom and configuration bits are sent for write by the firmware. If checksum error is detected by the firmware, the GUI retries 3 times. If all 3 tries fail, the write operation is aborted.

Boot loader upgrade procedure

This is not recommended and support is not guaranteed.

This guide assumes you have the ds30 Loader installed in the default location. The default location may differ between different firmware versions. Follow these steps carefully. The most critical step is 16: if it fails the boot loader is likely to stop working.

1. Make a copy of boot loader MPLAB project directory.
2. Open the MPLAB project of the temporary boot loader created in the previous step
3. Open settings.inc
4. Disable goto protection if available
5. Enable boot loader protection if available
6. Change the boot loader placement, BLPLP (in older versions this define may be located in ds30loader.s/asm):

Device family	**Boot loader placement of temporary boot loader, BLPLP/BLPLR**
PIC16F	Upgrade is not supported
PIC18F	BLPLP × 2 + 1
PIC18FJ	BLPLP + 2
PIC24F	BLPLR × 2 + 1
PIC24FJ	BLPLP + 2
PIC24H	BLPLP + 2
dsPIC30F	BLPLR × 2 + 1
dsPIC33FJ	BLPLP + 2

7. Build the project, Menu Project->Make or F10
8. Open ds30 Loader GUI and choose the hex file generated in the previous step
9. Write the temporary boot loader as you would with any hex file
10. If not already activated, activate advanced mod on the menu View->Advanced mode.
11. Check “Don’t write goto at 0x00” under the advanced tab
12. Check “Custom boot loader” under the advanced tab
13. Put value of BLPLP/BLPLR from the temporary boot loader the placement textbox
14. Put value of BLSIZEP/BLSIZE from temporary boot loader in the size textbox
15. Choose the hex file of the new boot loader
16. Make sure the temporary boot loader is active, either using time or use a led to indicate that the temporary boot loader is active.
17. Write
18. The boot loader upgrade is finished

Troubleshooting

Known issues

- With the Lawicel CAN adapters read operation is not possible for most combinations of bit rate and serial interface baud rate. Typically any bit rate above 50k will not work for read flash operation. Write operation still works.
- The IXXAT ISB-to-CAN compact adapter takes several hundred milliseconds to setup. This means that the firmware timeout cannot be set shorter than this if a reset command is used and the application uses a different bit rate than the boot loader.

Error and warning messages

Trouble	Possible cause	Solution
Verification failed	Broken chip	Replace the chip.
	Worn out flash	Replace the chip.
	Chip is used outside specifications	Read data sheet and make sure all specifications are met.
	Code protection is used	If code protection is used the device should be completely erased before the boot loader is written.
Uart rx buffer not empty as expected (is device sending data?)	This could happen if the device is not restarted properly and is sending data	Restart the device before pressing download.
Hardware detected a framing error	This might happen when the baud rate error is just on the limit.	Try a different baud rate, higher or lower. Needs to be changed in both the boot loader firmware and the host application.
		Try another com-port.
The hex-file contains code that will overwrite the boot loader	This occurs when the loaded hex-file contains memory locations that could overwrite the boot loader. It can also occur if the wrong device is selected.	Select correct device.
		Set custom boot loader properties under the advanced tab to the values from the firmware, usually called BLPLP and BLSIZEP in settings.inc.
		Free up program memory if needed.

	Because the application does not know the actual boot loader size prior to communication with the boot loader it assumes a size. This value might be wrong; one can tell the application the correct value under the advanced tab.	Reserve boot loader memory space in the application linker script.
Hex-file contains more config locations than the device has	If you use the export function from MPLAB this might happen.	Use the hex-file produced by the assembler or linker.
Last row containing configs was found in hex file, last page has been disabled. Consult manual for more information.	On PIC18FJ, PIC24FJ, and on some PIC24E/dsPIC33E devices the configuration words are stored as the last words in the ordinary flash address space. The configuration words are vital to the PIC operation. To avoid corrupting the configuration words or changing them which could lead to the boot loader not functioning properly the last page in flash will not get written.	To write the last page in flash, check the write configs checkbox under the advanced tab. It is unlikely that the last page will contain any code, therefore it is usually safe to not write the last page and this is also recommended.
There is no GOTO at location 0x00.	A custom boot loader is used where the application reset vector has been relocated.	Ignore the warning.

Appendix A – Links

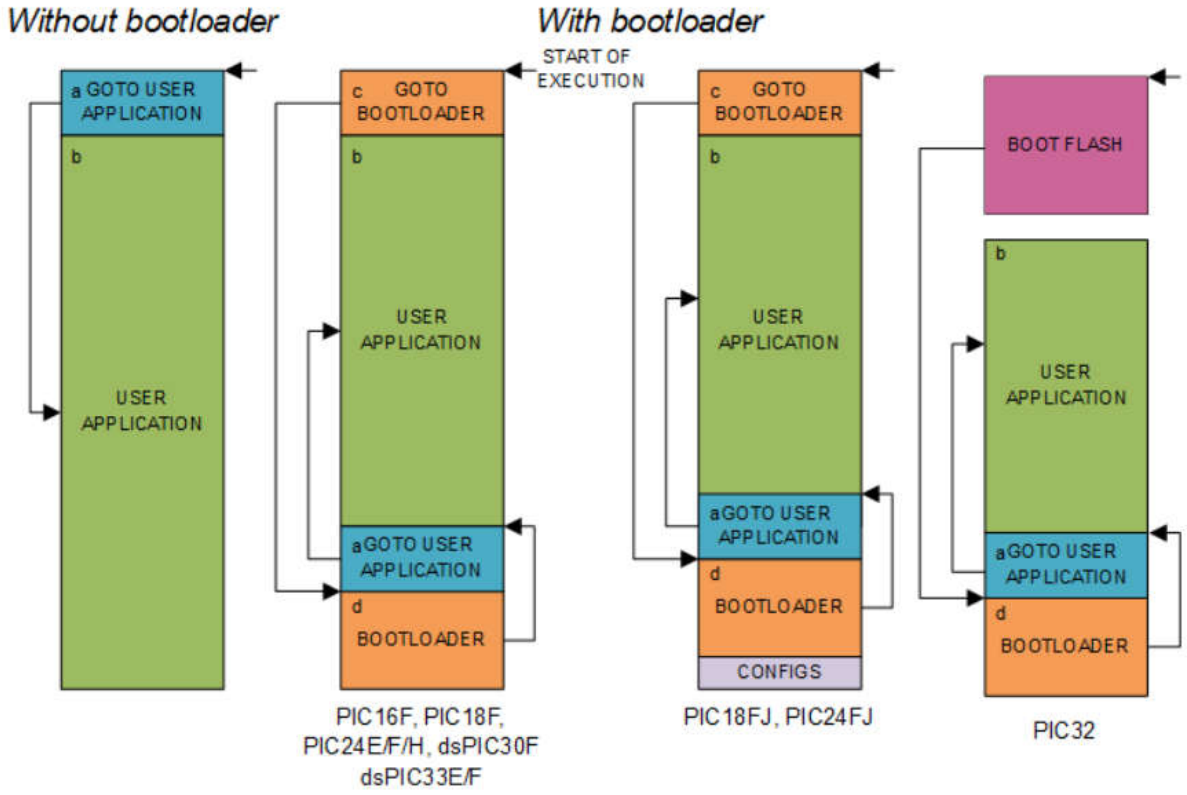
ds30 Loader website
https://ds30loader.com

ds30 Loader free edition website
https://picbootloader.com

Mono / MonoDevelop
http://www.mono-project.com

Appendix B – memory map

a. The goto application word at 0x00 is normally created automatically by the compiler. The GUI moves the goto to just before the boot loader code.
b. The user application is not affected by the boot loader
c. A new goto at 0x00 pointing to the boot loader code is created by the GUI.
d. The boot loader code is normally placed at the end of the memory.



Appendix C – Write time samples

These are sample times and should not be seen as a warranty, performance may be different in your application. Data is actual data that ends up in the flash memory.

UART	data [kB]	baud rate	time [s]	kB/s	Device
PIC12F	3,1	115 200	5,5	0,6	PIC12F1822
PIC16F	14,0	115 200	16,5	0,8	PIC16F1936
PIC18F	31,5	115 200	8,3	3,8	PIC18F2550
PIC18FJ	14,0	115 200	2,8	5,0	PIC18F24J11
PIC24F	15,5	115 200	2,3	6,7	PIC24F16KA102
PIC24FJ	125,0	115 200	15,5	8,1	PIC24FJ128GA010
PIC24H	126,5	115 200	15,8	8,0	PIC24HJ128GP504
PIC24E	510,0	115 200	57,9	8,8	PIC24EP512GU810
dsPIC30F	47,3	115 200	7,7	6,1	dsPIC30F4011
dsPIC33F	254,5	115 200	31,9	8,0	dsPIC33FJ256GP710
dsPIC33E	510,0	115 200	58,1	8,8	dsPIC33EP512MU810
PIC32MX	508,0	115 200	69,4	7,3	PIC32MX795F512L

CAN	data [kB]	bit rate	time [s]	kB/s	Device
PIC18F	61,6	250 000	15,1	4,1	PIC18F66K80
PIC24H	126,5	250 000	40,0	3,0	PIC24HJ128GP504
PIC24E	510,0	250 000	46,5	11,0	PIC24EP512GU810
dsPIC30F	47,0	250 000	5,9	8,0	dsPIC30F4011
dsPIC33F	253,0	250 000	84,0	3,0	dsPIC33FJ256GP710
dsPIC33E	510,0	250 000	45,8	11,1	dsPIC33EP512MU810
PIC32MX	508,0	250 000	51,0	10,0	PIC32MX795F512L

Appendix D - Files

Package content

File	Description
\copying.txt	The license for ds30 Loader
\bin	**Contains binaries**
\bin\canlibCLSNET.dll	Use by the Kvaser CAN port plug-in
\bin\PCANBasic.dll	Used by the PCAN CAN port plug-in
\bin\PCANBasic_x64.dll	Used by the PCAN CAN port plug-in, this needs to be manually renamed to PCANBasic.dll on x64 systems.
\bin\PCANBasic_x86.dll	Used by the PCAN CAN port plug-in, this needs to be manually renamed to PCANBasic.dll on x86 systems.
\bin\vcinet2.dll	Used by the IXXAT CAN port plug-in
\bin\vxlapi.dll	Used by the Vector CAN port plug-in
\bin\vxlapi64.dll	Used by the Vector CAN port plug-in
\bin\vxlapi_NET20.dll	Used by the Vector CAN port plug-in Requires: vxlapi.dll
\bin\ds30LoaderGui.exe	Graphical user interface for ds30 Loader
\bin\ds30LoaderConsole.exe	Console application for ds30 Loader.
\bin\ixxatBitRates.xml	Contains CAN timings used by IXXAT CAN port plug-in for different bit rates
\bin\vectorBitRates.xml	Contains CAN timings used by IXXAT CAN port plug-in for different baud rates
\documentation	**Contains all documentation for the ds30 Loader**
\firmware xxx	**Contains a MPLAB IDE project for a single Microchip device family**
\firmware xxx\ds30loader.mcp	MPLAB IDE project file
\firmware xxx\src\devices.inc	Contains device constants
\firmware xxx\src\settings.inc	Contains boot loader settings that should be checked/changed before download
\firmware xxx\src\user_init.inc	User initialization and exit code
\firmware xxx\src\ds30loader.s/asm	Contains the boot loader code

Files created by ds30 Loader

Created files are usually placed in user home directory\ .ds30Loader

File	Description
settingsPortLawicel.xml	Settings saved by Lawicel CAN port plug-in

settingsPortSerial.xml	Settings saved by serial port plug-in
settingsPortVector.xml	Settings saved by Vector CAN port plug-in
settings.xml	GUI settings
recentfile*.xml	GUI settings for recent files

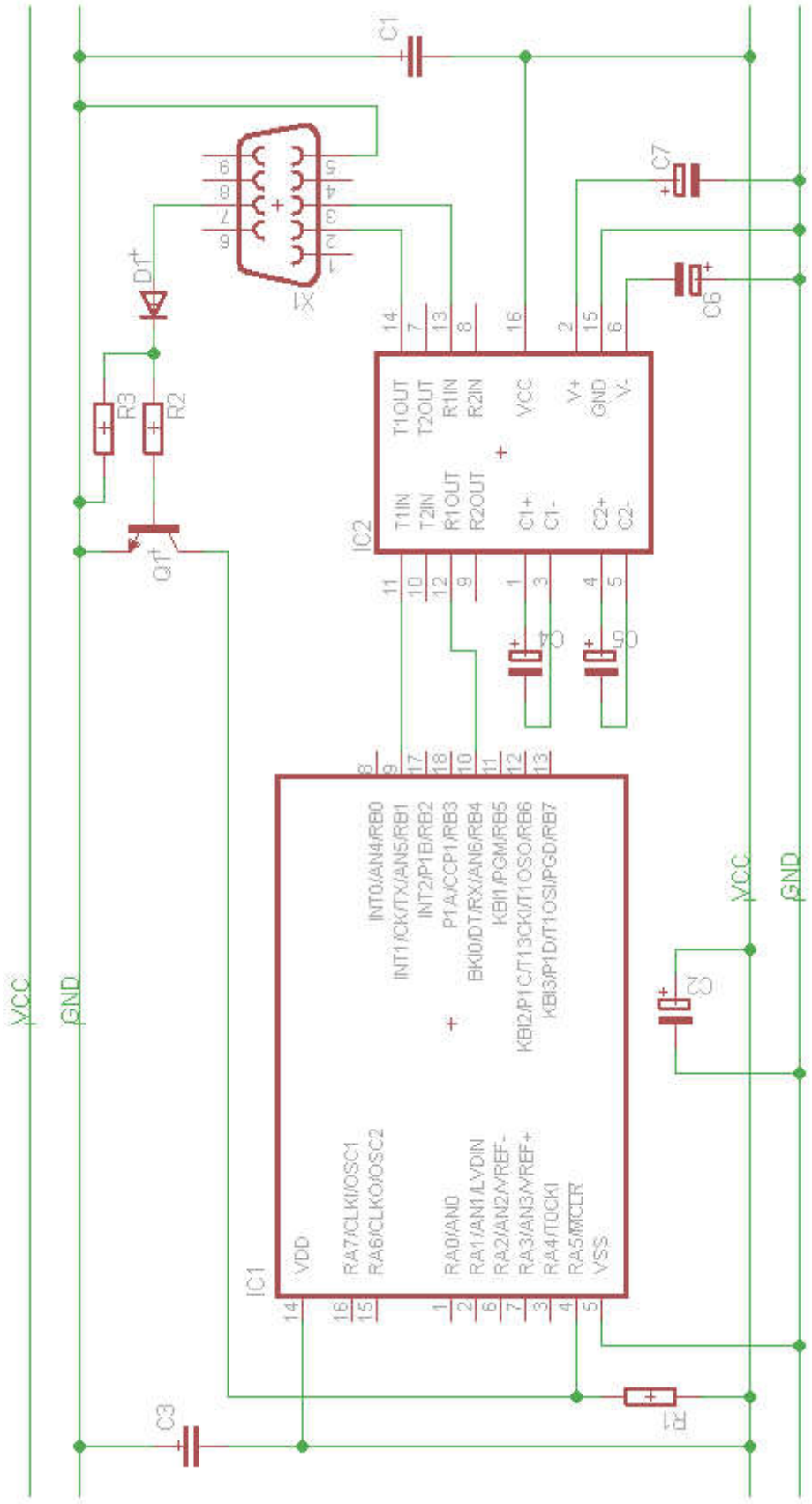
Appendix E - UART schematics

This is just a sample implementation of UART communication. The schematic comes without warranty.

Components

Name	Value		Comment
IC1	PIC	Required	PIC microcontroller
IC2	MAX202	Required	RS232 level converter
X1	DB9	Optional	Connector
R1	10kΩ	Required	Master clear pull-up
C1	100n	Recommended	Decoupling capacitor
C2	10µF	Recommended	Filter capacitor
C3	100nF	Recommended	Decoupling capacitor
C4	100nF	Required	Charge pump capacitor
C5	100nF	Required	Charge pump capacitor
C6	100nF	Required	Charge pump capacitor
C7	100nF	Required	Charge pump capacitor
R2	10kΩ	Optional	Reset by RTS, base current limiter
R3	10kΩ	Optional	Reset by RTS, base pull-down
D1	1N4148	Optional	Reset by RTS, RTS signal rectifier
Q1	BC547	Optional	Reset by RTS, RTS signal inverter

Schematic


C1
C7
C6
D1
R3
R2
Q1
IC2
T1OUT
T2OUT
R1IN
R2IN
VCC
V+
GND
V-
T1IN
T2IN
R1OUT
R2OUT
C1+
C1-
C2+
C2-
C5
C4
IC1
VDD
RA7/CLKI/OSC1
RA6/CLKO/OSC2
RA0/AN0
RA1/AN1/LVDIN
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/MCLR
VSS
INT0/AN4/RB0
INT1/CK/TX/AN5/RB1
INT2/P1B/RB2
P1A/CCP1/RB3
BKI0/DT/RX/AN6/RB4
KBI1/PGM/RB5
KBI2/P1C/T13CKI/T1OSO/RB6
KBI3/P1D/T1OSI/PGD/RB7
C2
C3
R1
VCC
GND

Appendix F – CAN schematics

This is just a sample implementation of CAN communication. The schematic comes without warranty.

Components

Name	Value		Comment
IC1	PIC	Required	PIC microcontroller
U1	MCP2551	Required	CAN transceiver
X1	DB9	Optional	Connector
R1	10kΩ	Required	Master clear pull-up
R2	120Ω	Optional	Bus termination
R3	22k Ω	Required	Slew rate control, see the datasheet
C2	10µF	Recommended	Filter capacitor
C3	100nF	Recommended	Decoupling capacitor

Schematic