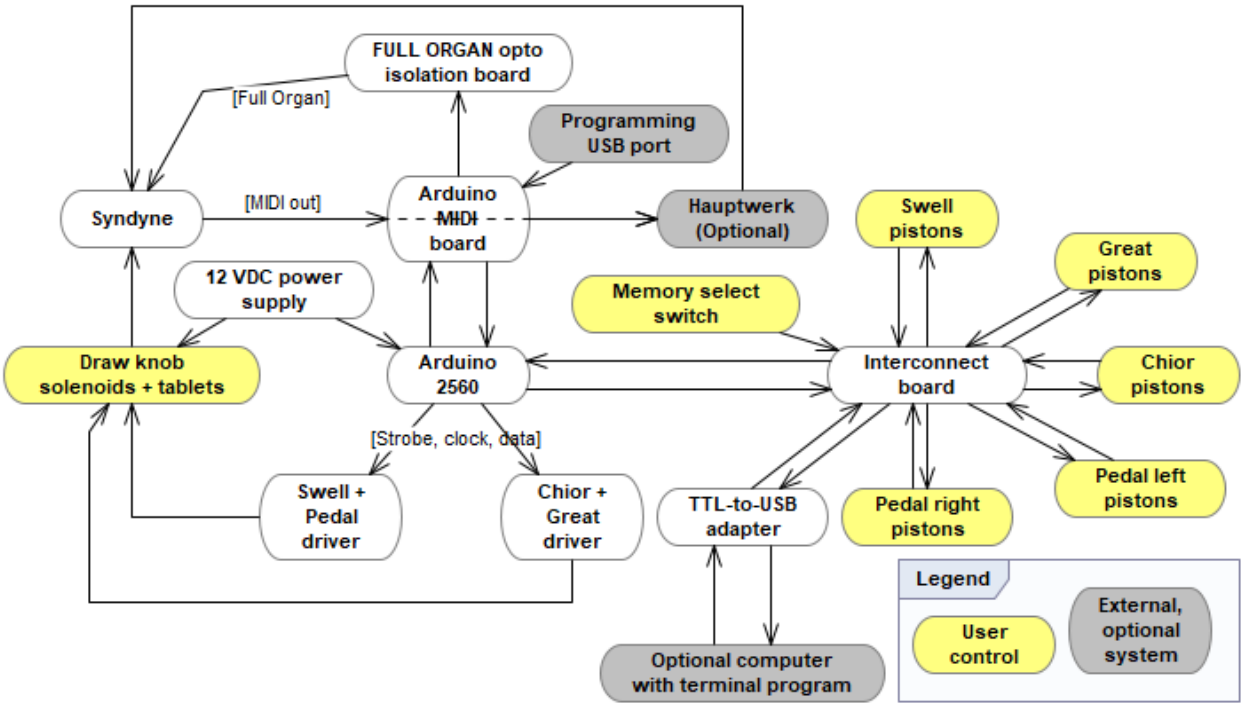
Organ Combination Action

Table of Contents

1 Introduction

The system comprises the following components:

- Syndyne (provides MIDI output)
- Arduino Mega 2560
- Arduino MIDI board (SparkFun DEV-12898 MIDI shield)
- Interconnect board (Pijl 2019)
- TTL-to-USB cable (for debug console)
- 12VDC power supply (powers solenoids & Arduino)
- Opto-isolator board (passes Full Organ signal to Syndyne)
- 2 solenoid driver boards (Swell + Pedal, Chior + Great)
- stop & tablet solenoids



1.1 Technical overview

The Arduino is continuously polling the pistons and reading MIDI notifications for stop changes (only).

When requested, it stores or retrieves values in its internal non-volatile memory. As needed, it will send new stop settings to the stop driver board(s).

1.2 System capacities

System area	Parameter	Limit	Used
Firmware: stop memory per division	maximum number of stops & couplers per division	32	

Hardware on GT & CH board	total number of stops & couplers on GT & CH	96	30 + 18
Division of GT & CH board	total number of stops and couplers on GT	40	30 wired
	total number of stops and couplers on CH	56 if fully populated	18 wired
CPU	EEPROM memory	4K	

Division	Console stops	Coupler tablets	Board outputs
Great	9	3	GT+CH board outputs 1-40
Chior	12	3	GT+CH board outputs 41-73 (even more channels can be populated)
Swell	15	3	SW+PD board outputs 1-40
Pedal	9	6	SW+PD board outputs 41-73 (even more channels can be populated)

9 generals * 16 bytes each = 144 bytes per level
4 div * 4 per div * 4 bytes each = 64 bytes per level
--> 208 bytes per level
4096 bytes has room for 20 levels

1.3 Inputs

Preset pistons

Division	Upper console	Lower console
Great	4	
Swell	4	
Chior	4	
Pedal	4	
(General)	4	5 + 4 + FullOrgan + GT--> Pedal

Memory level selection switch.

SFZ is not monitored by firmware.

MIDI from Syndyne

Optional: USB debug terminal
Optional: USB programming port

Arduino board has a Reset button.

The MIDI shield board has buttons too but the firmware isn't reading them.

1.4 Outputs

5 outputs for reading piston rows.
Full-organ output to Syndyne and indicator LED.
Activation/deactivation outputs for stop & tablet solenoids.
Output to optional Hauptwerk.
Optional output to USB computer debugging terminal

1.5 Using debug terminal

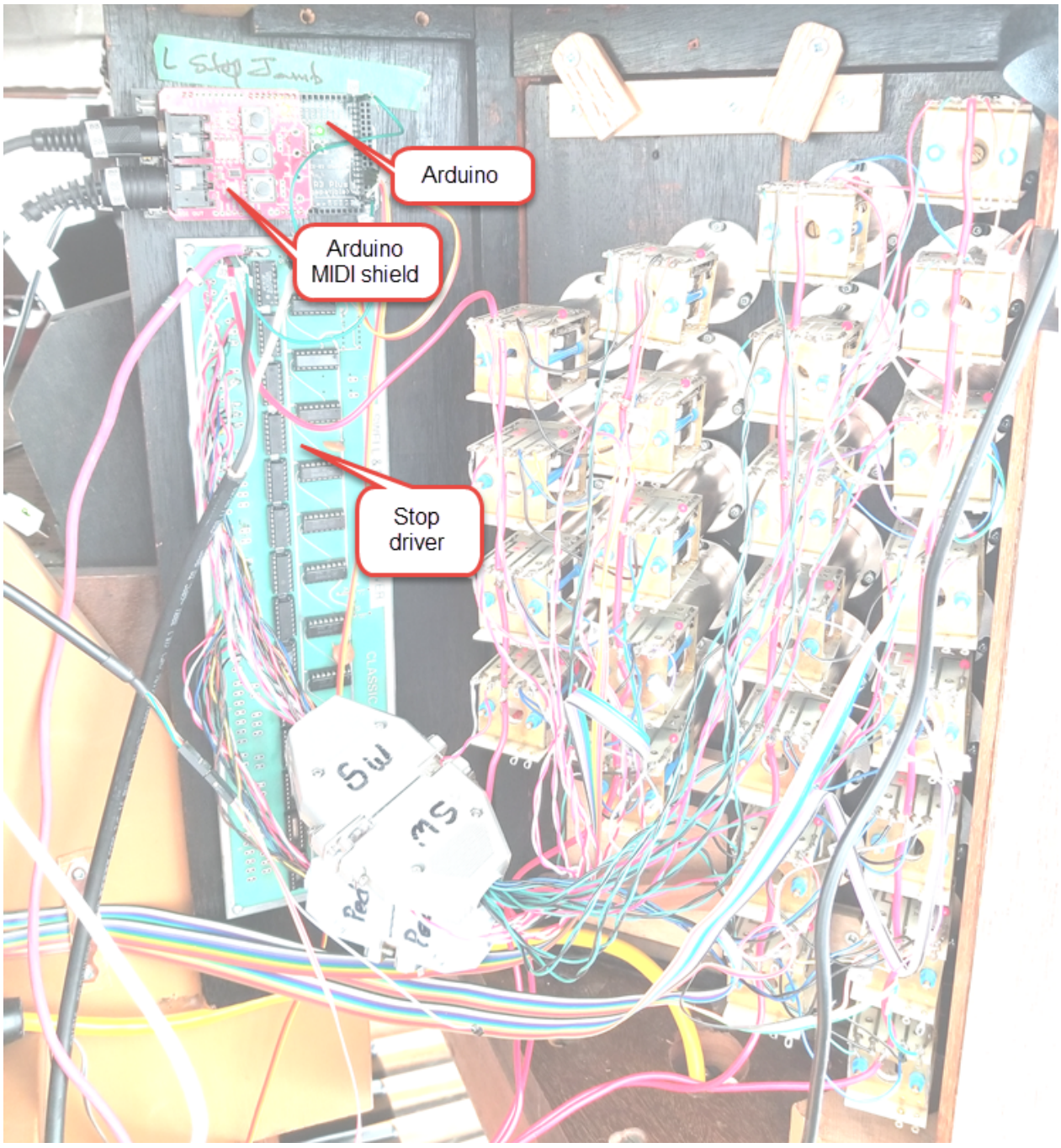
1. Connect the debug USB cable to the computer
2. Open a terminal program on the computer
3. Select the correct virtual COM port for the USB cable, and 19200, N81
4. Review output and/or enter commands

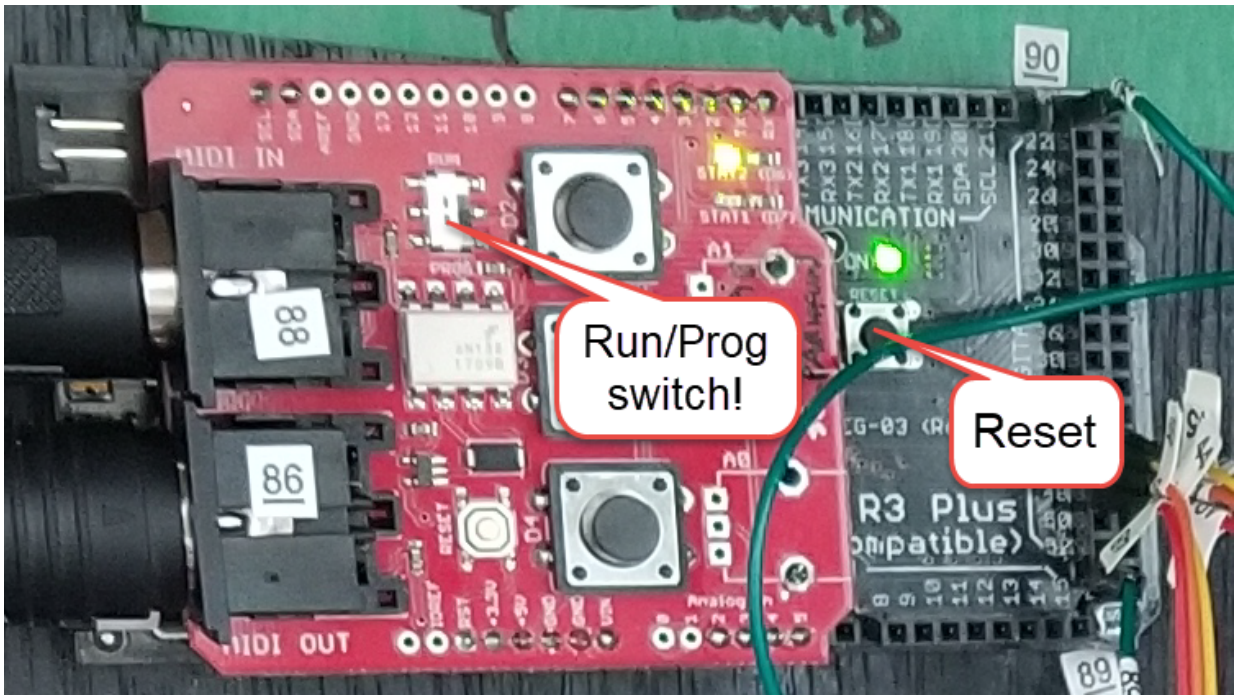
Indication: When the Arduino boots, it will say "Starting" in the terminal.

You can also issue commands to the firmware. When a command is recognized, the firmware will acknowledge the command. It could take a few seconds before the mode change request is honoured.

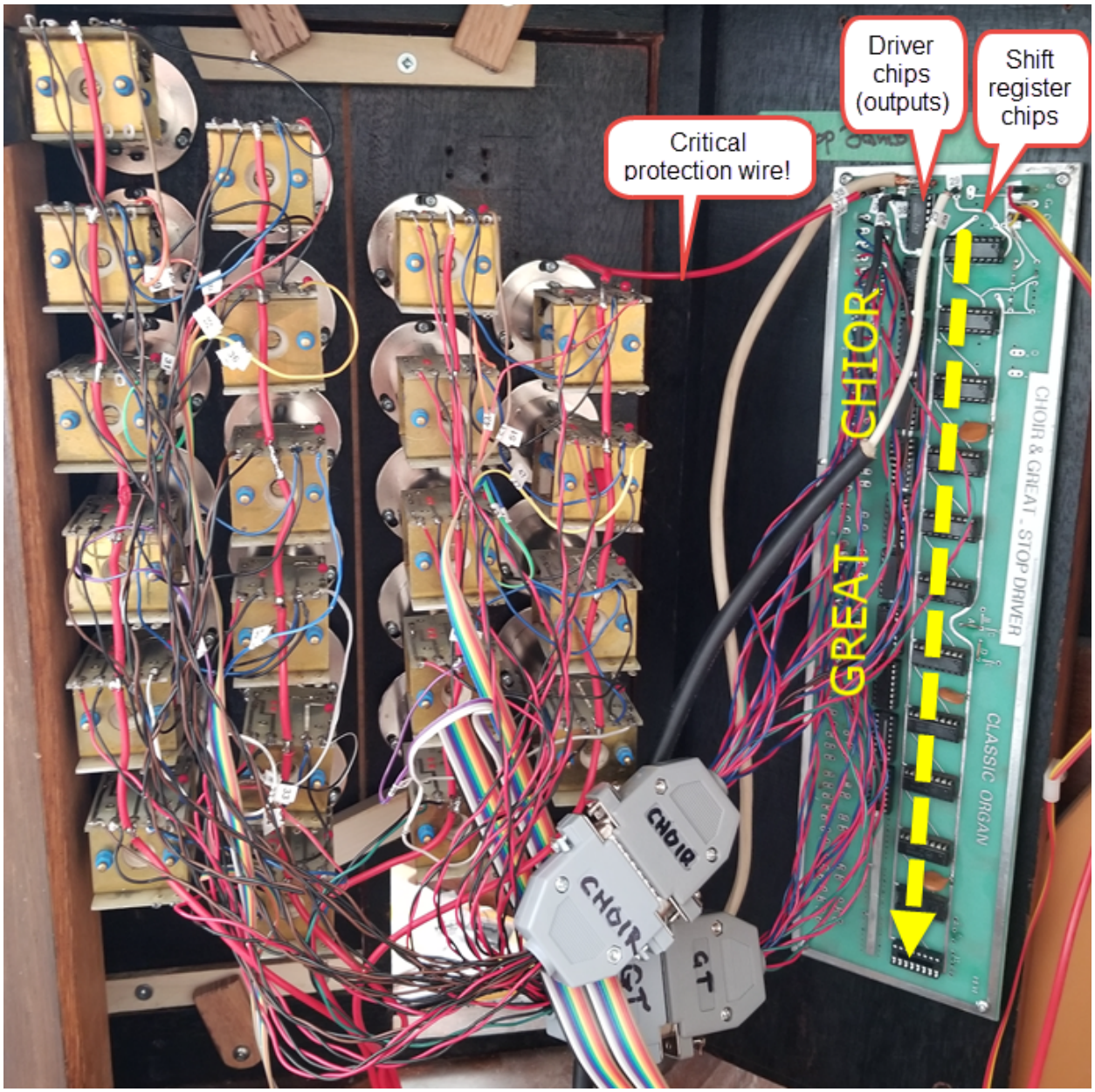
Command	Function
Normal	Activate normal operation
Custom	Activate custom test mode (see firmware)
Seq	Activate sequential test mode
CycleAll	Activate all soleniods on/off/on/off

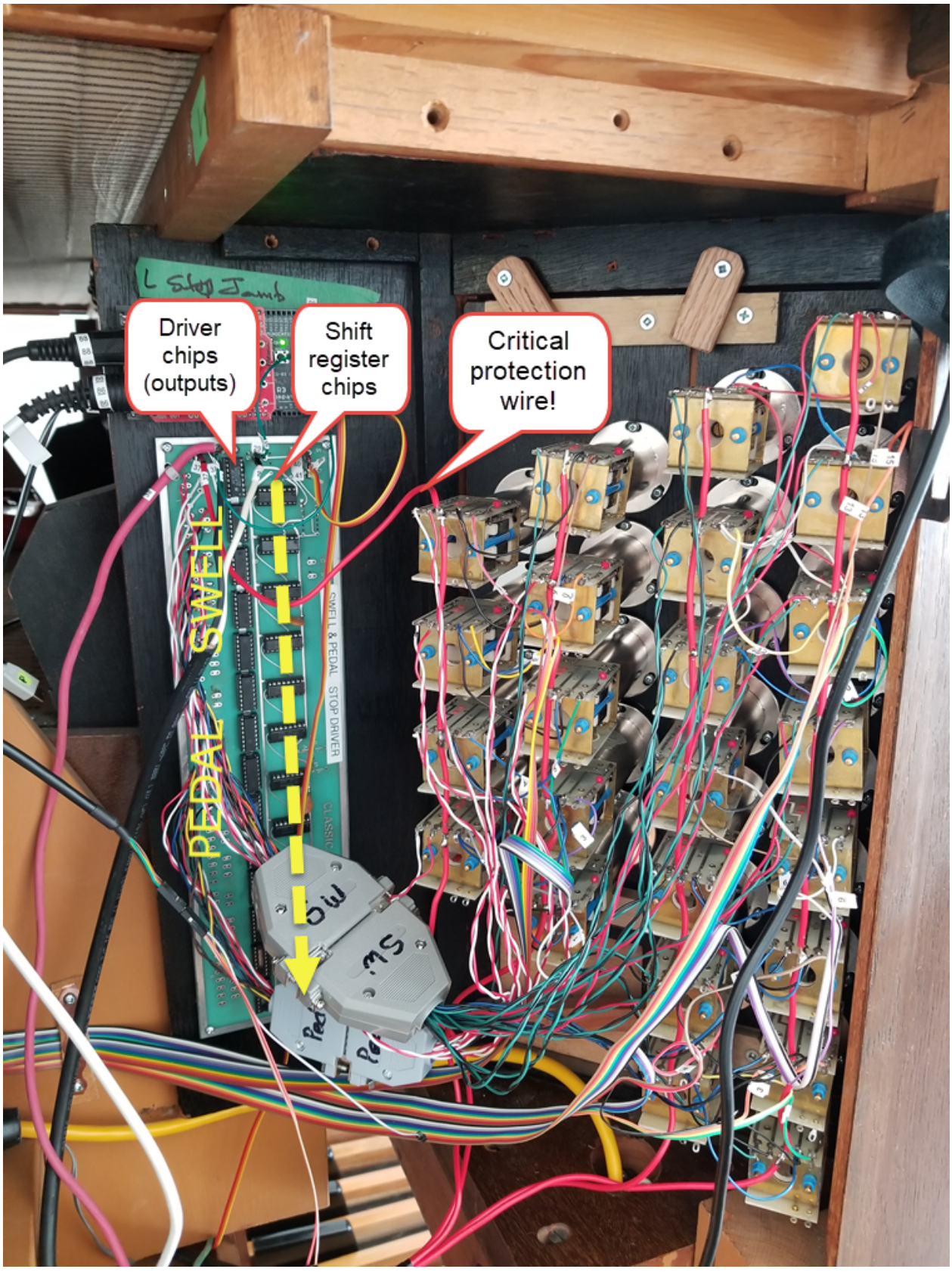
1.6 Anatomy





The dashed arrow shows the progression of data through the board. We can leave out driver chips we don't use. We can leave out shift register chips we don't use as well but because they are cascaded, we can only omit unused chips at the bottom end.





2 Theory of operation

Power for the combination action system is entirely provided by the 12 VDC supply at the left (west) end of the console.

Essentially, it polls the piston buttons and listens to the MIDI signals from Syndyne. When asked, it performs any of the following activities:

- General cancel
- Piston press
- Piston press with "Set" also pressed

When it needs to know the current stop settings, it uses the last-known stop/tablet positions notified via the MIDI interface from Syndyne.

Each division shares a stop-driver board (swell + pedal, great + chior). When it needs to set a divisional, the last-known tablet/stop positions for the "companion division" must be known so the "companion division" stop settings can be correctly asserted.

For example, since Swell & Pedal are companions together, recalling the Swell will require knowledge of the settings for Pedal since they're both set anytime we reload settings into the board.

Important: The Arduino can't "read" the stop settings. It only knows stop settings by observing the MIDI messages that Syndyne sends when it starts up or a change occurs.

The "memory" is inside the Arduino and requires no battery backup.

3 Firmware

The firmware is traditional Arduino C/C++.

Main classes:

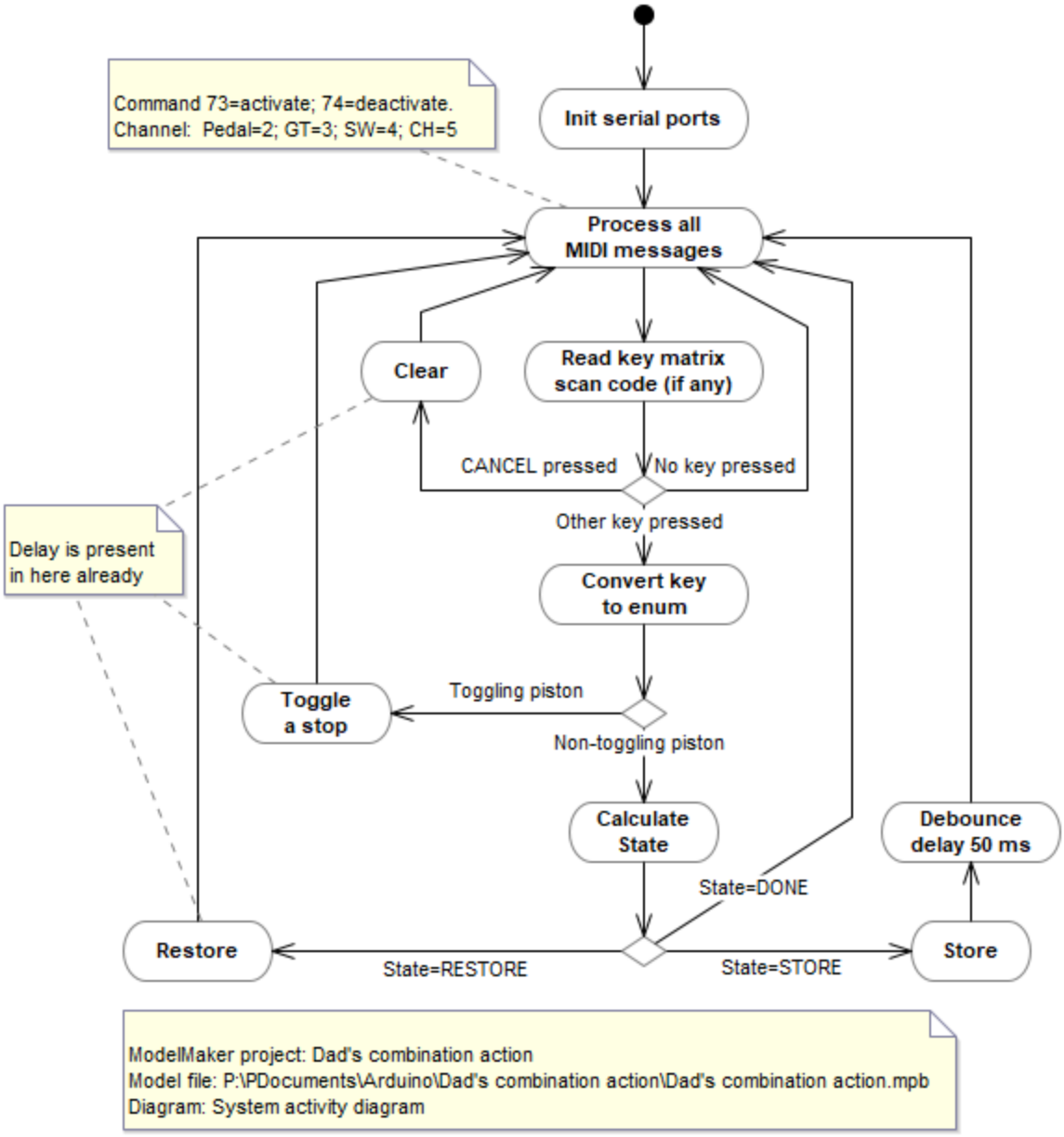
Class	Instances	Purpose
StopDriver	2 (SW+PD, GT+CH)	Drive stop soleniods to activate or deactivate a stop. Always fully reloads both division's outputs on board.
MidiReader	1	Read and interpret MIDI messages from Syndyne
StopState	1	Remember stop states according to received MIDI msgs

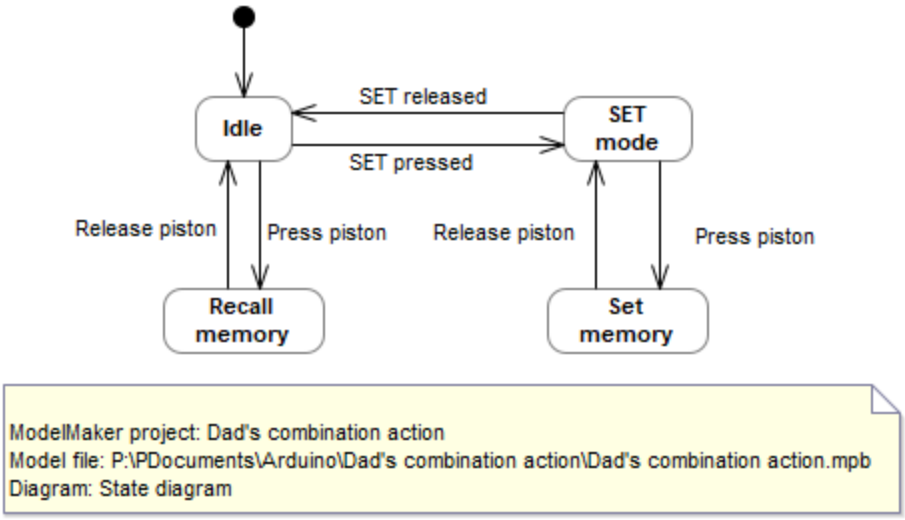
Firmware source code is stored in GitHub at:

https://github.com/MikeFromBC/Dad_CombinationAction

The firmware and the system diagram are the best indication of precisely how everything is connected.

3.1 Diagrams





3.2 Updating firmware

1. Connect to programming port, configure Arduino IDE port, board type
2. Switch MIDI board to PROG mode
3. Download program
4. Important: return MIDI board switch to RUN mode

4 Repair tips

All electronics are static-sensitive; use a grounding strap.

Stop driver board is not well supported physically. If removing/reinserting chips, be sure to carefully support the board with your hand during gentle reinsertion.

Both stop driver boards are basically identical although they're not populated identically.

The system relies on the piston buttons being reliable--especially the "Set" button.

5 Parts to have on hand

ULN2003A "High voltage, high current darlington transistor array".

6 Quirks

Because the firmware can't "read" the stop status, situations can arise where the firmware doesn't really know the status of the stops.

Consider this sequence of events:

1. Power-on organ
2. (syndyne boots & arduino boots)
3. (syndyne provides all current stop settings to Arduino via MIDI)
4. (now Arduino should know the stop settings at all times)
5. supposing the Arduino is reset for some reason (new firmware installed or firmware crashed)
6. (now Arduino has forgotten previous MIDI messages)

The problem begins on step 5.

You can stop the problem by using general cancel and activate the stops you wish to use.

7 Troubleshooting

The "debug terminal" is often a useful way to see what is happening.

"Set" button causes stops to move when a piston is pressed
This happens if the "Set" button isn't working or is not reliable.

A group of 3-4 stops not moving
Related driver chip may have failed. You could swap driver chips with a spare.

No stops moving on one side
A stop driver board may not be receiving all 3 connections from Arduino. Board may not be receiving power (comes from Arduino). Look very closely at the connections to the Arduino...are any pins backing out of the connector???? (all should go in the same amount.) If so, gently push them back in.

Lots of stops (but not all) not working on one side
Is the fault seemingly affecting everything after a certain stop? (eg: top half of Swell, all of Pedal?) That suggests a problem with one of the shift register chips or connections to it.

Nothing works
Confirm that the big 12 V DC power supply near the Arduino board is working. Confirm Arduino board(s) have at least one indicator on.

Is the MIDI shield's board switch set to "Run"? It should be.